VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY
INSTITUTE OF MATHEMATICS

OPTIMALIZACE VÝROBNÍCH PROCESŮ
OPTIMIZATION OF PRODUCTION PROCESSES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. David Halas**
AUTHOR

VEDOUCÍ PRÁCE — **RNDr. Pavel Popela, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav matematiky
Student:	**Bc. David Halas**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Matematické inženýrství
Vedoucí práce:	**RNDr. Pavel Popela, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Optimalizace výrobních procesů

Stručná charakteristika problematiky úkolu:

Diplomant si prohloubí své znalosti z teorie pravděpodobnosti, matematické statistiky a operačního výzkumu. Pro vybrané problémy z oblasti řízení výrobních procesů sestaví simulační modely a otestuje je. Dále bude aplikovat své znalosti matematického programování (celočíselného a stochastického) a vytvoří matematické modely pro vybrané rozhodovací problémy výrobního procesu, a to i při neurčitých parametrech a při rozhodování distribuovaném v čase. Programová implementace bude realizována s pomocí vhodných matematických softwarů a testována na reálným problémům blízkých datech. Výsledky práce budou využity v rámci projektu NETME+ a při reálných aplikačních úlohách.

Cíle diplomové práce:

1. Diplomant v práci nejprve uvede shrnutí souvisejících poznatků operačního výzkumu.
2. K vybrané problematice diplomant sestaví původní simulační model a popíše jej.
3. Dále diplomant uvede související optimalizační problémy a jejich matematické modely.
4. Podle vlastností uvedených modelů diplomant navrhne vhodné algoritmy řešení.
5. Algoritmy budou implementovány v matematickém software a prezentovány společně s testovacími výsledky.

Seznam literatury:

KLAPKA, Jindřich, Jiří DVOŘÁK a Pavel POPELA. 2001. Metody operačního výzkumu. Vyd. 2. Brno: VUTIUM. ISBN 8021418397.

ZIPKIN, Paul H. et al. 1993. Logistics of Production and Inventory. Handbook in Operations Research and Management Science, vol. 4, Elsevier. ISBN 978-0444874726.

HAX, Arnoldo C. a Dan CANDEA. 1984. Production and inventory management. Englewood Cliffs: Prentice-Hall. ISBN 0137248806.

GOSAVI, Abhijit. 2015. Simulation-Based Optimization. Springer Verlag. ISBN 978-1489974914.

WILLIAMS, H. Paul. 2013. Model building in mathematical programming. 5th ed. Hoboken, N.J.: John Wiley & Sons. ISBN 978-1118443330.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2016/17

V Brně, dne

L. S.

prof. RNDr. Josef Šlapal, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Tato práce se zabývá modelováním různých typů výrobních linek. Modelování je prováděno pomocí matematického programování a simulačního přístupu. Optimalizační výpočty jsou většinou realizovány s pomocí programu GAMS. Simulace jsou realizovány programem Matlab/ SimEvents. Výsledky jsou prezentovány Ganttovými diagramy.

Summary

This thesis deals with modelling diffenrent types of production lines. Modeling is done by the mathematical programming and simulation methods. Optimization related computations are mostly implemented in program GAMS. Simulation is realized by using program Matlab/SimEvents. The results are presented by the Gantt diagrams.

Klíčová slova

celočíselná optimalizace, matematické programování, simulace výroby, heuristická optimalizace, rozvrhování

Keywords

integer optimization, mathematical programming, simulation of production, heuristic optimization, scheduling

HALAS, D. Optimalizace výrobních procesů. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 71 s. Vedoucí diplomové práce RNDr. Pavel Popela, Ph.D..

Prohlašuji, že jsem diplomovou práci *Optimalizace výrobních procesů* vypracoval samostatně pod vedením RnDr. Pavla Popely, Ph.D. s použitím materiálů uvedených v seznamu literatury.

Bc. David Halas

Chtěl bych poděkovat svému vedoucímu RnDr. Pavlu Popelovi, Ph.D. za odborné vedení a rady při zpracování této diplomové práce. Také bych chtěl především poděkovat Ing. Tomášovi Mauderovi, Ph.D. za cenné rady, pomoc a zkušenosti.

Obsah

Úvod

První část práce je zaměřena na matematické programování výrobních linek a jejich implementaci. Výrobní linky lze rozdělit na sériové, paralelní a jejich kombinace, které nazýváme obecnými výrobními linkami. V praxi se často setkáváme s obecnými výrobními linkami. U těchto linek se dají zkoumat některé charakteristiky například tvoření front nebo optimální plánování údržeb na strojích. V této práci jsou uvedeny obecné i konkrétní matematické modely. Práce s těmito modely je prezentována na příkladech a je k nim uvedena jejich výpočtová náročnost. Modely jsou implementovány v programu GAMS a výsledky ilustrovány Ganttovými diagramy v MS EXCEL. Další část práce je zaměřena na simulační přístup, který využívá program SimEvents, což je část programu Matlab-Simulink. Tento text se dělí na 4 kapitoly.

První kapitola popisuje základní pojmy a definice, které spadají do oblasti operačního výzkumu a budou použity v dalších kapitolách. Také je tu pojednáno o Ganttových diagramech včetně vysvětlení jejich použití v práci.

Druhá kapitola je hlavní částí práce. Úvod je zaměřen na pojmy matematického programování. Podkapitoly se zabývají druhy výrobních linek, sestavením jejích matematického modelu a využitím na příkladu a ukázkou výsledků na Ganttových diagramech. Druhy výrobních linek, kterými se kapitola zabývá, jsou sériové, paralelní a obecné výrobní linky. Obecná výrobní linka je potom rozšířena o fronty a o plánování údržeb.

Třetí kapitola navrhuje přístup simulační a také představuje program SimEvents. Tímto přístupem jsou řešeny příklady z druhé kapitoly. Výsledky jsou porovnány také s výsledky z druhé kapitoly. Dále se kapitola zabývá heuristikou a algoritmem, který řeší plánování údržeb.

Závěrečná kapitola shrnuje výpočetní náročnost matematických modelů a simulací. Tyto náročnosti jsou srovnány a zhodnoceny.

Práce může být užitečná pro iniciativu Industry 4.0. Jedním z cílů tohoto trendu je digitalizace a automatizace lidských činností v oblasti výroby pro přesnější a rychlejší práci. Jedním ze způsobů má být zajištění předávání informací mezi výrobky pomocí čipů na každém výrobku a materiálu. Vize Industry 4.0 je obsáhlá a její součástí je například vize o virtuální továrně, o které pojednává článek [16] a [30].

Kapitola 1

Základní pojmy a formát výstupních dat

V této kapitole uvedeme a vysvětlíme základní pojmy, které budeme využívat v celé práci. Dále také popíšeme způsob grafického znázorňování výsledků. Nejprve než si vysvětlíme, co je to výrobní linka, zavedeme pojmy entita, stroj a procesní čas.

Definice 1 *Entitou nazýváme obecnou jednotku. Příklady entit v různých oblastech uvádí tabulka. V této práci budeme entitou myslet výrobek nebo součástku.*

Tabulka 1.1: Příklady entit v různých oblastech

Oblast aplikace	**Entity**
Letiště	Letadla čekající na přistání
Komunikační sítě	Odesílané pakety, zprávy
Eskalátory	Cestující lidé
Výrobní linky	Výrobky, součástky
Počítačový operační systém	Úkoly

Definice 2 *Strojem nazýváme proces, kdy je prováděna činnost na entitě po určitou dobu. Tuto dobu nazýváme procesním časem. Příkladem stroje může být obráběcí stroj, pokladna, člověk apod. Entity do stroje vstupují a zase z něho vycházejí po uplynutí procesního času. Stroje značíme písmenem M a pořadovým číslem.*

Definice 3 *Výrobní linkou nazýváme posloupnost strojů. Výrobní linku lze rozdělit na tři části. Vstupní částí pro entity je počáteční stroj, navazující částí je výrobní proces, který zajišťují opět stroje a poslední částí je koncový stroj, ze kterého entity vystupují hotové. Tento popis ilustruje obrázek 1.1.*

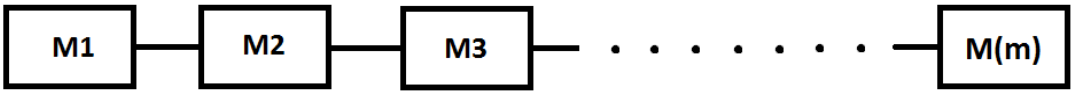
Obrázek 1.1: Schéma výrobní linky

Jedno z významných jmen, které se pojí s výrobními linkami, je Henry Ford. Tento muž zavedl pásovou výrobu a rozdělil ji na mnoho dílčích procesů na začátku 20. století. Více se dá dočíst v [29]. My se budeme zabývat výrobními linkami pouze z matematického hlediska a modelovat je. Ukázkou výrobní linky může být ukázka 1.

Ukázka 1 *Výroba pístů do motorů: na začátek výrobní linky se nám dopravují nařezané hliníkové válce(entita), které v lisu (1. stroj) pod vysokou teplotou jsou slisovány do prvotní podoby. Následně jsou dopraveny do pece (2. stroj), kde jsou vystaveny vysokým teplotám pro získání lepších vlastností. Poté jsou dopraveny do obráběcího stroje(3. stroj), kde jsou válce soustruženy, frézovány a do nich vyvrtány potřebné díry. Posledním krokem je očištění, vybroušení a umytí (4.-6. stroj). Výrobní linka má tedy 6 strojů.*

Podle typologie výrobních linek můžeme rozlišit dva základní druhy. Tím prvním je výrobní linka zapojena sériově. Tento typ budeme dále nazývat sériovou linkou a tuto linku můžeme schématicky vidět na obrázku 1.2. Více o sériové lince pojednává kapitola 2.1.

Definice 4 *Sériovou linkou nazýváme posloupnost strojů zapojených za sebou. Entity musí projít všemi stroji v pořadí v jakém jsou stroje zapojeny.*

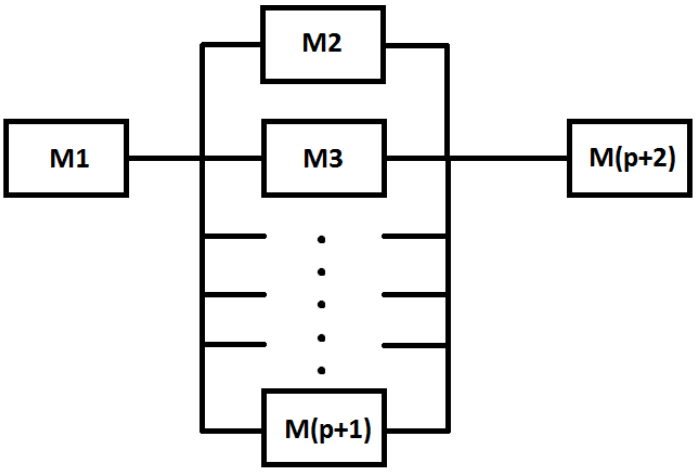


Obrázek 1.2: Sériová výrobní linka

Druhý typ výrobní linky budeme nazývat paralelní linkou. Obrázek 1.3 ilustruje tento typ, který bude popsán v kapitole 2.2.

Definice 5 *Paralelní linkou nazýváme výrobní linku, jejíž stroje jsou zapojeny paralelně. Entita projde právě jedním strojem.*

Častým typem výrobní linky je kombinace sériové i paralelní, kterou budeme nazývat obecnou výrobní linkou. O tomto typu pojednává kapitola 2.4. Ve stejné kapitole budeme obecnou výrobní linku převádět na orientovaný graf.

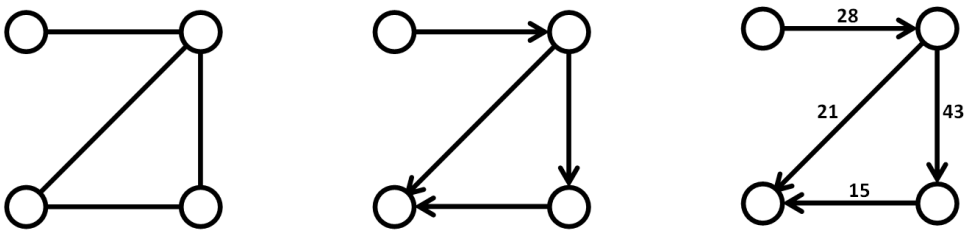


Obrázek 1.3: Paralelní část výrobní linky

Definice 6 *Grafem G nazýváme uspořádanou dvojici $G = (V, E)$, kde $V \neq \emptyset$ je množina vrcholů a E je množina hran. Pro neorientovaný graf platí, že E je podmnožinou všech možných dvojic vrcholů.*[14]

Definice 7 *Orientovaným grafem nazýváme graf, jehož množina E je podmnožinou všech uspořádaných dvojic vrcholů. Zapisujeme $E \subseteq V \times V$, kde $V \times V$ označuje kartézský součin prvků z množiny V. Hrany orientovaného grafu nazýváme orientované hrany a graficky je označujeme šipkami.*[14]

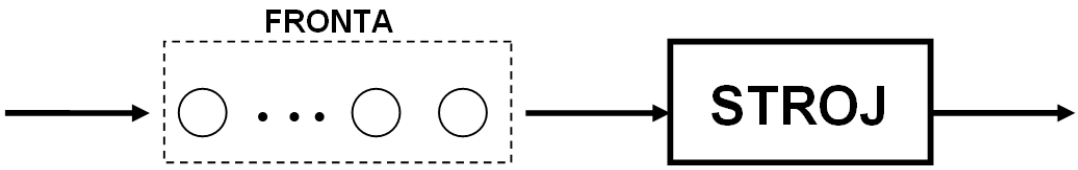
Definice 8 *Ohodnocením hran nazveme funkci w, která hraně přiřadí číslo w: $E \to \mathbb{R}$, kde $\mathbb{R}$ je množina všech reálných čísel. Pro nezáporné ohodnocení hran platí $w(e) \geq 0$, kde $e \in E$.*[14]



Obrázek 1.4: vlevo: neorientovaný graf; uprostřed: orientovaný graf; vpravo: orientovaný graf s ohodnocením hran

U výrobní linky se často zkoumá, jestli se tvoří fronty a jak velké. S jednoduchou frontou se můžeme například setkat v obchodě. Zákaznici tvoří čekající řadu a jsou obsluhováni poté, co je pokladna volná. Ve výrobní lince se fronty tvoří stejně, avšak může

záležet na pravidlech odchodu entit z fronty. Na obrázku 1.5 vidíme jednoduché schéma fronty. V praxi se snažíme tvoření front minimalizovat. O práci s frontami pojednává kapitola 2.5. Rozlišujeme několik základních pravidel pro odchod entit z front.



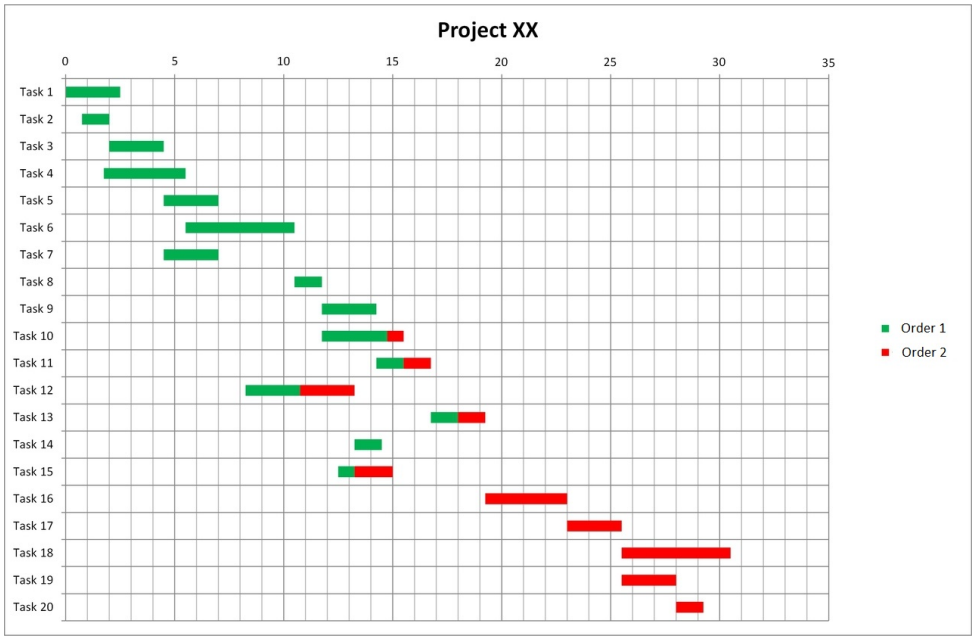
Obrázek 1.5: Schéma fronty

- FIFO (first in, first out) - První entita, která vstupuje do fronty, také z fronty jako první vystupuje. Příkladem je řazení entit za sebou.
- LIFO (last in, first out) - Poslední entita, která do fronty vstupuje, tak jako první z fronty vystupuje. Příkladem je skládání entit na sebe.
- SIRO (selection in random order) - Entity vystupují z fronty v náhodném pořadí.
- GD (general discipline) - Libovolné pravidlo.

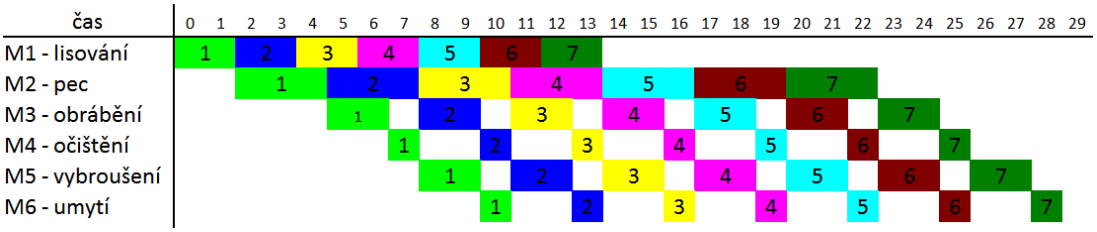
Další oblastí, kterou budeme zkoumat, jsou údržby.

Definice 9 *Údržbou nazýváme proces na stroji s dobou trvání, kdy do stroje nevstupuje ani z něj nevystupuje entita. Tuto dobu budeme nazývat doba údržby. Cílem údržby je především kontrola stroje nebo jeho oprava.*

K prezentaci výsledků vycházejících z modelů budeme využívat Ganttovy diagramy, o kterých pojednává zdroj [6]. Ganttův diagram je speciální pruhový diagram pojmenovaný po H. L. Ganttovi. Ukázku Ganttova diagramu vidíme na obrázku 1.6. Často se využívá k řízení činností, které jsou naplánovány v čase. Potřebnými údaji pro vytvoření tohoto diagramu jsou počáteční a koncové časy činností. Na vodorovné ose se nachází časová osa. Na svislé ose jsou potom jednotlivé činnosti. My budeme Ganttův diagram vytvářet pomocí programu MS Excel, avšak tento program nenabízí vytváření těchto diagramů ve svých funkcích. Proto musíme využít jazyka VBA (Visual Basic for Applications) a diagramy doprogramovat. O použití jazyka VBA v programu MS Excel se dozvíme v [32]. Na obrázku 1.7 vidíme Ganttův diagram ukázky 1 pro 7 hliníkových válců (entit). Na svislé ose jsou zadány stroje. Entity jsou očíslovány a barevně rozlišeny. Tento diagram byl vytvořen pomoci jazyka VBA v aplikaci MS Excel.



Obrázek 1.6: Ukázka Ganttova diagramu, zdroj [15]



Obrázek 1.7: Ganttův diagram ukázky 1

Kapitola 2

Model výrobní linky - matematické programování

V kapitole 1 jsme si uvedli základní pojmy, které se týkají výrobní linky. Také jsme popsali základní druhy výrobních linek. V této kapitole sestavíme matematické modely sériové, paralelní a obecné výrobní linky pomocí matematického programování. Dále modely rozšíříme o fronty a údržby. Každý model zařadíme do jedné z úloh matematického programování LP, NLP, MIP nebo MINLP. Tyto pojmy úvodem vysvětlíme.

Každý matematický model se skládá z účelové funkce a omezujících podmínek. Tyto modely můžeme rozdělit podle charakteru podmínek a účelové funkce na lineární a nelineární. Modely dále dělíme podle charakteru proměnných na celočíselné, neceločíselné nebo smíšené. Neceločíselnými máme na mysli, že proměnné jsou z oboru reálných čísel.

Formulace minimalizační úlohy

$$
\begin{array}{ll}
\text{Účelová funkce:} & \min f(x_1, \ldots, x_N) \\
\text{Podmínky:} & h_i(x_1, \ldots, x_N) \leq 0, \quad \text{pro } i = 1 \ldots, K \\
& g_i(x_1, \ldots, x_N) = 0, \quad \text{pro } i = 1 \ldots, L \\
& x_1, \ldots, x_N \in X
\end{array}
$$

,kde $f,\ h_1, \ldots,\ h_K,\ g_1, \ldots,\ g_L$ jsou funkce definované na $\mathbb{R}^N$. Proměnné $x_1, \ldots,\ x_N$ jsou hledaná řešení a X je množina. Pokud podmínky s nerovnostmi "$\leq$"upravíme vynásobením číslem -1, dostáváme podmínky typu "$\geq$", viz [18] a [4].

Lineární programování LP (linear programming) je úloha, kde účelová funkce f a funkce h, g, w jsou zadány jako lineární kombinace neznámých $x_1, \ldots,\ x_N$ s konstantními koeficienty. Jestliže jedna z těchto funkcí je nelineární funkcí, pak mluvíme o nelineárním programování NLP (nonlinear programming). Obsáhlejší definici najdeme v [18].

O programování s celými čísly IP (integer programming) mluvíme tehdy, platí-li $X \subseteq \mathbb{Z}$, kde $\mathbb{Z}$ je množina celých čísel. Pro programování s reálnými čísly platí $X \subseteq \mathbb{R}$, kde $\mathbb{R}$ je množina reálných čísel. Programování se smíšenými čísly MIP (mixed-integer programming) nazýváme úlohu, která pracuje s celými i reálnými čísly, tedy s kombinací

předchozích možností. Pokud se bude navíc jednat o nelineární úlohy, tak budeme značit MINLP (mixed-integer nonlinear programming). O těchto úlohách pojednává zdroj [18].

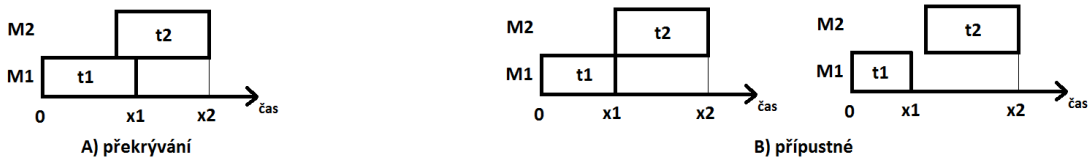
2.1 Sériová výrobní linka

V této kapitole se budeme zabývat sériovou výrobní linkou podle definice 4. Tento druh výrobní linky lze jednoduše matematicky modelovat a výpočet patří k méně náročným. Výrobní linka může být celá sériově poskládaná nebo alespoň nějakou sériovou část obsahovat. Výrobek se chová tak, že je zpracován na stroji a po dokončení je určitým způsobem přesunut do dalšího stroje. Tento proces se opakuje. Výrobek je tedy v určitém pořadí opracováván na všech strojích sériové výrobní linky. Výrobek nemůže žádný stroj vynechat, přeskočit nebo si volit jiné pořadí strojů. Strojům budeme přiřazovat index j a entitám index i. Budeme zanedbávat dopravníky mezi stroji. To znamená, jakmile entita skončí ve stroji j, ihned může začít být zpracovávána na stroji $(j+1)$.

Mějme m strojů spojených sériově. Chceme, aby linkou prošlo n entit. Označme t_j jako procesní čas na stroji j, kde $j = 1 \ldots,\ m$. Budeme prozatím předpokládat, že zpracování každé entity trvá na stroji j stejně. Dále potřebujeme proměnnou $x_{i,j}$, jejíž hodnota znamená čas konce zpracování entity i na stroji j. Mezi každými dvěma spojenými stroji bude platit nerovnice (2.1), která je uvedena ve zdroji [33].

$$x_{i,j+1} - x_{i,j} \geq t_{j+1}, \text{ kde } i = 1, \ldots,\ n,\ j = 1, \ldots,\ m-1 \tag{2.1}$$

Nerovnici (2.1) popisuje obrázek 2.1, který vysvětluje, že překrývající procesy A) nesmí nastat. Přípustné jsou pouze procesy, které navazují ihned nebo později, jak na obrázku ilustruje situace B).



Obrázek 2.1: Překrývání procesů mezi stroji

Takhle jsme určili pořadí strojů pro každou entitu. Tímto určením budeme říkat, že jsme entitám zadali cestu přes výrobní linku. Nyní je potřeba uspořádat procesy entit na každém stroji j. Použijeme stejné nerovnice, které trochu pozměníme.

$$x_{i+1,j} - x_{i,j} \geq t_j, \text{ kde } i = 1, \ldots,\ n-1,\ j = 1, \ldots,\ m \tag{2.2}$$

Nerovnice (2.2) zajišťují nepřekrývání procesů na každém stroji. Je tedy požadováno, aby se procesy nepřekrývaly jak mezi stroji, tak na jednotlivých strojích. Také díky těmto nerovnicím se nemůže stát, že by se entity předběhly. To znamená, že pokud si entity očíslujeme a zpracují se na prvním stroji v pořadí $1, 2, 3, 4, 5 \ldots$, tak se ve stejném pořadí zpracují na každém stroji. Vlastnost nepředbíhání požadujeme v sériových částech výrobní linky.

Pro celý model předpokládáme, že $x_{i,j}$ jsou nezáporná reálná čísla. Nyní máme hotové podmínky modelu. Abychom mohli model dokončit, potřebujeme účelovou funkci. Máme více možností, jak tuto funkci volit a záleží na tom, jak potřebujeme, aby se model choval. Požadujeme, aby entity byly co nejdříve zpracovány na každém stroji a nedocházelo k mezerám mezi procesy na jednotlivých strojích. K těmto mezerám může dojít pouze, pokud se tvoří fronty. Chceme potom minimalizovat každou hodnotu proměnných $x_{i,j}$, které jsou nezáporné. Tohoto docílíme minimalizováním součtu všech hodnot $x_{i,j}$. Vztah (2.3) vyjadřuje naši účelovou funkci.

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j} \tag{2.3}$$

Tím je sestaveno vše potřebné pro modelování sériové výrobní linky. Celá sériová výrobní linka není moc zajímavá z hlediska použití, avšak dobře ukazuje na základní vlastnosti výrobních linek a jejich matematický popis. Podmínky tohoto modelu nám budou pomáhat k sestavování obecných výrobních linek, jejichž části bývají sériové. Nakonec uvádíme tabulku shrnutí celého modelu 2.1 sériové výrobní linky a její klasifikaci. Z důvodu přehlednosti budeme v tabulkách shrnutí oddělovat vzorce a podmínky čarami jak vertikálně tak horizontálně.

Tabulka 2.1: Model sériové výrobní linky

Shrnutí modelu-sériová výrobní linka	
$\min \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j}$	
$x_{i,j+1} - x_{i,j} \geq t_{j+1}$	$i = 1, \ldots, n,\ j = 1, \ldots, m-1$
$x_{i+1,j} - x_{i,j} \geq t_j$	$i = 1, \ldots, n-1,\ j = 1, \ldots, m$
$x_{i,j} \geq 0$	$i = 1, \ldots, n,\ j = 1, \ldots, m$

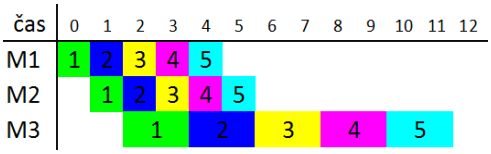
Na tomto shrnutí můžeme vidět, že matematické programování sériové výrobní linky je úlohou lineárního programování. Tyto úlohy jsou lehce řešitelné a mohou být počítány simplexovou metodou, která je vysvětlena v [18], případně efektivnějšími metodami.

Příklad 1

Zadání příkladu ilustruje obrázek 2.2. Na obrázku vidíme 3 stroje, které jsou zapojeny sériově. Dalším zadáním příkladu je, že budeme počítat s 5-ti entitami a procesní časy volíme $t = (1, 1, 2)$, pro jednoduchou prezentaci výsledků. Implementování úlohy provedeme v programu GAMS a zdrojový kód najdeme v příloze pod názvem *seriova_linka.gms*.

Obrázek 2.2: Sériová výrobní linka se třemi stroji

Po vypočítání hledaných časů $x_{i,j}$, hodnoty uložíme do programu MS Excel, který nám vykreslí Ganttův diagram 2.3. Na diagramu vidíme, že procesy navazují podle očekávání a délky procesních časů odpovídají. Pořadí entit se taky nemění a procesy se nepřekrývají. Touto sériovou výrobní linkou prošlo 5 entit za 12 časových dílků.



Obrázek 2.3: Ganttův diagram-příklad sériové výrobní linky

2.2 Paralelní výrobní linka - nelineární model

V této části se budeme zabývat paralelní výrobní linkou podle definice 5. Spojení strojů paralelně se často hodí, pokud chceme zrychlit výrobu. Například v obchodě máme více pokladen a stačí zákazníkovi použít pouze jednu, zatímco na dalších pokladnách mohou být obsluhování další lidé ve stejný čas. Jakmile se výrobek v lince dostane před soustavu paralelních strojů, musí se určitým způsobem rozhodnout, do kterého stroje půjde. Například v obchodě se rozhodujeme podle toho, která pokladna je volná. Ve výrobních linkách o rozřazení a dopravení rozhoduje počítač připojený na mechanickou dopravu (rameno, portál atd.). Tady budeme předpokládat, že způsob rozřazování entit do paralelních strojů je dáno nebo může být libovolné a my chceme model podle toho sestrojit. Způsobů rozřazování existuje několik a my budeme používat logiku střídání, kterou definujeme v průběhu této kapitoly. Další možností je ponechat volbu rozřazování volnou.

To znamená, že necháme model samotný spočítat, které rozřazování je vhodné. Poté vyhodnotíme, jestli se rozřazování chová podle logiky střídání.

Mějme p strojů spojených paralelně a jeden stroj na vstupu a jeden na výstupu, dohromady opět m strojů. Situaci vidíme na obrázku 1.3 z kapitoly 1.

Vstupní stroj M1 opracovává entity a posílá je do jednoho z paralelních strojů M2 až M(p+1). Až je proces entity v jednom z těchto strojů vyhotoven, tak je entita poslána na opracování do stroje M(p+2). Stroje M1 a M(p+2) je nutné do modelu zahrnout. Mohou to být stroje, které zakončují a začínají sériovou část, nebo je můžeme chápat jako vstupy a výstupy entit. Nejprve budeme model sestavovat podle základních poznatků o paralelních výrobních linkách pomocí nelineárního programování. Kvůli vyšším nárokům nelineárního programování poté model linearizujeme.

Začneme stejnými nerovnicemi jako v předchozí kapitole. Avšak budou platit pouze pro stroje M1 a M(p+2). Jsou to nerovnice (2.4), které vyjadřují vztahy mezi procesy entit i na strojích jednotlivých M1 a M(p+2).

$$x_{(i+1),j} - x_{i,j} \geq t_j,\ j = 1, m,\ i = 1, \ldots,\ n-1 \tag{2.4}$$

Nyní musíme popsat cestu entit. U paralelní výrobní linky může mít každá entita jinou posloupnost strojů, můžeme říkat jinou cestu. To je způsobeno tím, že každá entita, může jít přes jiný paralelní stroj. Proto bude daná situace o trochu složitější. Musíme si zavést tzv. indikátorové proměnné $d_{i,j}$ pro $i = 1, \ldots,\ n,\ j = 1, \ldots,\ m$, které nám pomohou pracovat s logickými podmínkami. Tyto proměnné jsou binární a definovány popisem (2.5). Indikátorové proměnné se zavádí ve zdroji [33].

$$d_{i,j} = \begin{cases} 1 & \text{pokud proces entity } i \text{ je prováděn na stroji } j \\ 0 & \text{pokud proces entity } i \text{ není prováděn na stroji } j \end{cases} \tag{2.5}$$

Každá entita může být zpracována pouze na jednom paralelním stroji. Musí proto platit logická podmínka (2.6) dovolující, aby pouze jedna indikátorová proměnná příslušná p paralelním strojům byla hodnoty 1.

$$\sum_{j=2}^{p+1} d_{i,j} = 1,\ i = 1, \ldots,\ n \tag{2.6}$$

Teď zajistíme cestu pro každou entitu podle hodnoty $d_{i,j}$. Nerovnice (2.7) a (2.8) jsou rozšířením nerovnic (2.1) a (2.2).

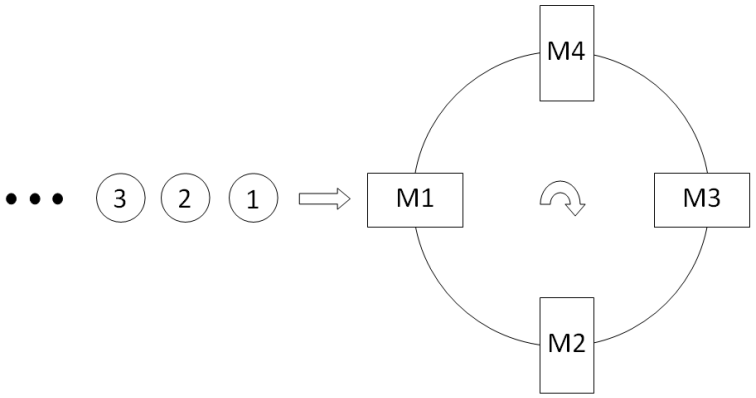
$$\sum_{j=2}^{p+1} d_{i,j} x_{i,j} - x_{i,1} \geq \sum_{j=2}^{p+1} d_{i,j} t_j,\ i = 1, \ldots,\ n \tag{2.7}$$

$$x_{i,p+2} - \sum_{j=2}^{p+1} d_{i,j} x_{i,j} \geq t_{p+2},\ i = 1, \ldots,\ n \tag{2.8}$$

Nerovnice (2.7) znamená, že pokud entita například půjde přes paralelní stroj 3, tzn. $d_{i,3} = 1$, pak ostatní indikátorové proměnné pro příslušnou entitu jsou nulové díky podmínce (2.6) a v nerovnici (2.7) zůstanou pouze členy $x_{i,3}$, $x_{i,1}$ a parametr t_3. Podobně platí pro nerovnici (2.8). V modelu s logikou střídání vynecháme použití proměnných $d_{i,j}$ pro indexy $j = 1, m$.

V této fázi má model potřebné základní vlastnosti, ale není ošetřeno, aby se procesy nepřekrývaly na paralelních strojích. Dalším krokem je doplnit vztahy mezi procesy entit na jednotlivých strojích. V tomto případě to není tak jednoduché. Můžeme situaci zkonkretizovat tak, že si určíme způsob přiřazování entit do paralelních strojů. Nejzákladnější technika přiřazování entit je logika střídání. To znamená 1. entita jde do 1.stroje, 2.entita jde do druhého atd., jakmile se dostaneme k poslednímu stroji, začínáme s přiřazováním zase od 1.stroje.

Definice 10 *Logikou střídání budeme nazývat proces, kdy každé entitě přiřazujeme stroj v pořadí 1 až p. Můžeme tedy říci, že se entity přiřazují v kružnici. Viz obrázek 2.4 pro 4 paralelní stroje.*



Obrázek 2.4: Schéma logiky střídání pro 4 paralelní stroje

Abychom použili logiku střídání, zavedeme pro tuto situaci nový index k . Definujeme procesy na paralelních strojích pomocí vztahu (2.9), za předpokladu, že p dělí n beze zbytku.

$$x_{j-1+(k+1)p,j} - x_{j-1+kp,j} \geq t_j,\ k = 0 \ldots,\ (n/p - 1),\ j = 2, \ldots,\ (p + 1) \tag{2.9}$$

Indikátorové proměnné $d_{i,j}$, budou speciálně voleny, aby byla dodržena logika střídání. Pro případ p=4 a m=6, zapíšeme $d_{i,j}$ pro lepší představu pomocí matice 2.10.

$$\{d_{i,j}\} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \; i = 1, \ldots, \; n, \; j = 2, \ldots, \; p \tag{2.10}$$

Tímto jsme model dokončili pro logiku střídání a s přidanou účelovou funkcí (2.3) uvedeme shrnutí.

Tabulka 2.2: Model paralelní výrobní linky s logikou střídání

Shrnutí modelu-paralelní výrobní linka s logikou střídání	
$\min \; \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j}$	
$x_{i+1,j} - x_{i,j} \geq t_j$	$j = 1, m, \; i = 1, \ldots, \; n-1$
$\sum_{j=2}^{p+1} d_{i,j} = 1$	$i = 1, \ldots, \; n$
$\sum_{j=2}^{p+1} d_{i,j} x_{i,j} - x_{i,1} \geq \sum_{j=2}^{p+1} d_{i,j} t_j$	$i = 1, \ldots, \; n$
$x_{i,p+2} - \sum_{j=2}^{p+1} d_{i,j} x_{i,j} \geq t_{p+2}$	$i = 1, \ldots, \; n$
$x_{j-1+(k+1)p,j} - x_{j-1+kp,j} \geq t_j$	$k = 0 \ldots, \; (n/p-1), \; j = 2, \ldots, \; (p+1)$
$d_{j-1+kp,j} = 1$	$k = 0 \ldots, \; (n/p-1), \; j = 2, \ldots, \; (p+1)$
$x_{i,j} \geq 0, \; d_{i,j} \in \{0, 1\}$	$i = 1, \ldots, \; n, \; j = 1, \ldots, \; m$

V modelu se vyskytují nelineární podmínky a celočíselné a reálné proměnné. Proto úlohu označíme jako MINLP.

Může se stát, že budeme chtít řešit model tak, že chceme nechat volnost výběru paralelních strojů pro každou entitu. Takže budeme chtít optimalizovat přiřazování entit strojům. Potom musíme použít obecnější vztahy pro entity na paralelních strojích. To znamená vystihnout nerovnicemi všechny možnosti přiřazení pro všechny entity. To už není tak jednoduché, protože s rostoucím počtem entit bude růst i počet možností.

Ukážeme, jak by situace vypadala pro 5 entit a 2 paralelní stroje podle obrázku 2.5.

$$d_{(i+1),j}x_{(i+1),j} - d_{i,j}d_{(i+1),j}x_{i,j} \geq d_{(i+1),j}t_j, i = 1, \ldots, \ 4; \ j = 2, 3 \tag{2.11}$$

Nerovnice (2.11) udává vztahy mezi entitami na stroji j, pokud entity ihned navazují.

$$d_{3,j}x_{3,j} - d_{3,j}d_{1,j}(1 - d_{2,j})x_{1,j} \geq d_{3,j}(1 - d_{2,j})t_j, \ j = 2, 3 \tag{2.12}$$

Nerovnice (2.12) platí pro to, když entita 3 a 1 byla zahrnuta do stroje j, ale entita 2 nebyla. A podobně až pro 5 entit jsou dány vztahy (2.13) ... (2.17).

$$d_{4,j}x_{4,j} - d_{4,j}d_{2,j}(1 - d_{3,j})x_{2,j} \geq d_{4,j}(1 - d_{3,j})t_j, \ j = 2, 3 \tag{2.13}$$

$$d_{4,j}x_{4,j} - d_{4,j}d_{1,j}(1 - d_{2,j})(1 - d_{3,j})x_{1,i} \geq d_{4,j}(1 - d_{2,j})(1 - d_{3,j})t_j, \ j = 2, 3 \tag{2.14}$$

$$\begin{aligned} d_{5,j}x_{5,j} - d_{5,j}(1 - d_{4,j})(1 - d_{3,j})(1 - d_{2,j})d_{1,j}x_{1,j} \geq \\ \geq d_{5,j}(1 - d_{4,j})(1 - d_{3,j})(1 - d_{2,j})t_j, \ j = 2, 3 \end{aligned} \tag{2.15}$$

$$d_{5,j}x_{5,j} - d_{5,j}(1 - d_{4,j})(1 - d_{3,j})d_{2,i}x_{2,i} \geq d_{5,j}(1 - d_{4,j})(1 - d_{3,j})t_j, \ j = 2, 3 \tag{2.16}$$

$$d_{5,j}x_{5,j} - d_{5,j}(1 - d_{4,j})d_{3,j}x_{3,j} \geq d_{5,j}(1 - d_{4,j})t_j, \ j = 2, 3 \tag{2.17}$$

Pro více entit bychom museli přidat další nerovnice, které by popisovaly další možnosti. V tomto zjednodušení může nastat 32 možností, jak mohou být entity rozřazeny. Obecně tedy můžeme spočítat pro obecný případ, že bude platit n^p možností. Kde n je počet entit a p počet paralelních strojů. Možnosti by se potom opět zvýšily, kdybychom měli v celé lince více řad paralelních strojů. Takže pro k řad by ještě platilo kn^p. Bylo by dobré zkusit nerovnice (2.11),... (2.17) zapsat obecněji pro n entit, například pomocí matic. Mohli bychom potom vidět nějaké pravidlo v matici koeficientů soustavy a potom jednodušeji sestavovat tyto nerovnice.

Dále musíme poznamenat, že tento model díky vztahům (2.11),... (2.17) se stává nelineárním. S tím potom mohou být problémy při výpočtech. Také může být obtížnější pracovat s tímto modelem pro větší výrobní linku s více řadami paralelních strojů a více entitami. Tady jsme se omezili pouze na jednu paralelní část. V praxi bývá pro rozřazování entit do strojů nějaké naprogramované pravidlo, jako například zmíněná logika střídání. O konkrétní časové náročnosti pojednává kapitola 4.1. Shrnutí modelu platí pro obecný počet paralelních strojů.

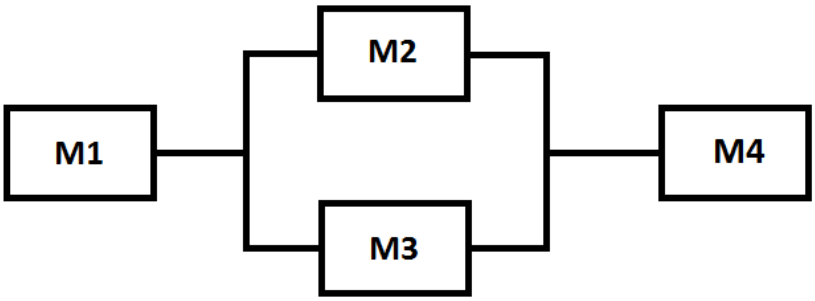
Tabulka 2.3: Model paralelní výrobní linky pro 5 entit bez pravidla přiřazování

Shrnutí modelu-paralelní výrobní linka pro n=5	
$\min \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j}$	
$x_{i+1,j} - x_{i,j} \geq t_j$	$j = 1, m,\ i = 1, \ldots,\ n-1$
$x_{1,1} \geq t_1$	
$\sum_{j=2}^{p+1} d_{i,j} = 1$	$i = 1, \ldots,\ n$
$\sum_{j=2}^{p+1} d_{i,j} x_{i,j} - x_{i,1} \geq \sum_{j=2}^{p+1} d_{i,j} t_j$	$i = 1, \ldots,\ n$
$x_{i,p+2} - \sum_{j=2}^{p+1} d_{i,j} x_{i,j} \geq t_{p+2}$	$i = 1, \ldots,\ n$
$d_{i+1,j} x_{i+1,j} - d_{i,j} d_{i+1,j} x_{i,j} \geq d_{i+1,j} t_j$	$i = 1, \ldots,\ 4,$ $j = 2, \ldots,\ (p+1)$
$d_{3,j} x_{3,j} - d_{3,j} d_{1,j} (1 - d_{2,j}) x_{1,j} \geq d_{3,j} (1 - d_{2,j}) t_j$	$j = 2, \ldots,\ (p+1)$
$d_{4,j} x_{4,j} - d_{4,j} d_{2,j} (1 - d_{3,j}) x_{2,j} \geq d_{4,j} (1 - d_{3,j}) t_j$	$j = 2, \ldots,\ (p+1)$
$d_{4,j} x_{4,j} - d_{4,j} d_{1,j} (1 - d_{2,j})(1 - d_{3,j}) x_{1,j} \geq$ $d_{4,j} (1 - d_{2,j})(1 - d_{3,j}) t_j$	$j = 2, \ldots,\ (p+1)$
$d_{5,j} x_{5,j} - d_{5,j} (1 - d_{4,j})(1 - d_{3,j})(1 - d_{2,j}) d_{1,j} x_{1,j} \geq$ $d_{5,j} (1 - d_{4,j})(1 - d_{3,j})(1 - d_{2,j}) t_j$	$j = 2, \ldots,\ (p+1)$
$d_{5,j} x_{5,j} - d_{5,j} (1 - d_{4,j})(1 - d_{3,j}) d_{2,j} x_{2,j} \geq$ $d_{5,j} (1 - d_{4,j})(1 - d_{3,j}) t_j$	$j = 2, \ldots,\ (p+1)$
$d_{5,j} x_{5,j} - d_{5,j} (1 - d_{4,j}) d_{3,j} x_{3,j} \geq d_{5,j} (1 - d_{4,j}) t_j$	$j = 2, \ldots,\ (p+1)$
$x_{i,j} \geq 0,\ d_{i,j} \in \{0, 1\}$	$i = 1, \ldots,\ n,\ j = 1, \ldots,\ m$

Model klasifikujeme jako úlohu s nelineárními podmínkami s celočíselnými a reálnými proměnnými. Patří tedy do skupiny MINLP.

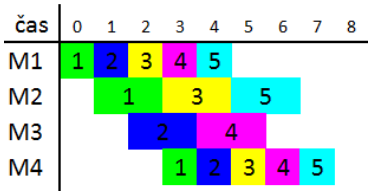
Příklad 2

Mějme zapojení podle obrázku 2.5. Máme 4 stroje z toho 2 paralelní a budeme počítat s 5 entitami. Na tento příklad aplikujeme logiku střídání pro paralelní stroje. Procesní časy volíme t=(1, 2, 2, 1).



Obrázek 2.5: Příklad: n=5, p=2, m=4

Po implementaci v programu GAMS dostaneme Ganttův diagram 2.6. Vidíme, že se entity ve strojích M2 a M3 chovají podle definované logiky střídání. Touto logikou jsme dosáhli toho, že 5 entit projde výrobní linkou za 8 časových dílků. Model je opět zahrnut v příloze pod názvem *para_nolin_strid.gms* .

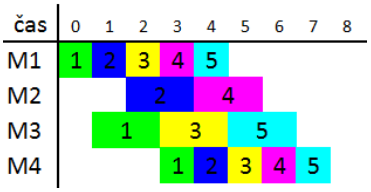


Obrázek 2.6: Ganttův diagram-logika střídání

Příklad 3

Zadání příkladu je opět podle obrázku 2.5 se stejnými procesními časy. Tentokrát však nemáme určenou logiku pro přiřazování entit do paralelních strojů. Implementování jsme provedli v programu GAMS a zdrojový kód najdeme v příloze pod názvem *para_nolin.gms*. Výsledný Ganttův diagram ilustruje obrázek 2.7. Na diagramu vidíme, že rozřazování entit v paralelních strojích se chová podle logiky střídání, jenom stroje byly voleny opačně.

Z toho můžeme usoudit, že volba této logiky je zde optimální. Dostali jsme stejný závěr, že 5 entit projde výrobní linkou za 8 časových dílků.



Obrázek 2.7: Ganttův diagram-bez definované logiky

2.3 Paralelní výrobní linka - lineární model

Paralelnímu modelu však chybí ještě jedna vlastnost. V paralelních částech může být dovoleno předbíhání entit. Jinak v sériových částech výrobní linky tomu tak být nesmí. A dále bychom chtěli pracovat s lineárním modelem. Linearizováním modelu docílíme toho, že model budeme moct snadněji použít i pro více entit. Model budeme sestavovat podle obrázku 1.3.

Začneme tím, že nerovnicemi popíšeme cestu každé entity. Nerovnice (2.18) definují cestu entity i ze stroje M1 do jedno z paralelních strojů M2 až M(p+1). Binární proměnná $d_{i,j}$ je definovaná předpisem (2.5). Nerovnice (2.19) popisuje cestu entity i z paralelních strojů M2 až M(p+1) do stroje M(p+2). Protože pro každou entitu i musí být zvolen pouze jeden paralelních stroj, tak pro $d_{i,j}$ platí podmínka (2.20).

$$x_{i,j} - x_{i,1} \geq t_j d_{i,j},\ i = 1, \ldots,\ n,\ j = 2, \ldots,\ (p+1) \tag{2.18}$$

$$x_{i,p+2} - x_{i,j} \geq t_{p+2} d_{i,j},\ i = 1, \ldots,\ n,\ j = 2, \ldots,\ (p+1) \tag{2.19}$$

$$\sum_{j=1}^{p+1} d_{i,j} = 1,\ i = 1, \ldots,\ n \tag{2.20}$$

Protože se nám bude hodit vědět i časy začátku procesů entit i na strojích j, zavedeme proměnnou $s_{i,j}$, kde $i = 1, \ldots, n$, $j = 1, \ldots, m$. Tuto proměnou použijeme v rovnici (2.21). Tato rovnice také říká, že pokud entita i nepůjde přes stroj j, tak se časy začátků a konců procesů budou rovnat. To stačí k tomu, aby tyto hodnoty neovlivňovaly model, když s nimi nechceme počítat. Rovnice však budou platit vždy na prvním a posledním stroji, takže musíme doplnit podmínky (2.22) a (2.23).

$$s_{i,j} + t_j d_{i,j} = x_{i,j},\ i = 1, \ldots,\ n,\ j = 1, \ldots,\ m \tag{2.21}$$

$$d_{i,1} = 1,\ i = 1, \ldots,\ n \tag{2.22}$$

$$d_{i,p+2} = 1,\ i = 1, \ldots,\ n \tag{2.23}$$

Tímto bychom měli vše k sestavení modelu pro paralelní výrobní linku. Další vlastností, kterou jsme požadovali, bylo předbíhání entit. Zadefinujeme binární proměnnou $e_{i,l,j}$, kde $i = 1, \ldots,\ n,\ l = 1, \ldots,\ n,\ i \neq l$ danou zápisem (2.24).

$$e_{i,l,j} = \begin{cases} 1 & \text{pokud proces entity } i \text{ předchází proces entity } l \text{ na stroji } j \\ 0 & \text{pokud proces entity } l \text{ předchází proces entity } i \text{ na stroji } j \end{cases} \tag{2.24}$$

Aby se nestalo, že by proces entity i předcházel proces entity l a zároveň naopak, přidáme podmínku (2.25), která dovoluje, aby platila pouze jedna situace.

$$e_{i,l,j} + e_{l,i,j} = 1,\ i = 1, \ldots,\ n,\ l = 1, \ldots,\ n,\ i \neq l,\ j = 1, \ldots,\ m \tag{2.25}$$

Nerovnice (2.26) propojují binární proměnné $e_{i,l,j}$ a proměnné $x_{i,j}$ a $s_{i,j}$. Dále je lze vysvětlit takto:
Jestliže $e_{i,l,j} = 1$, pak platí $x_{i,j} \leq s_{l,j}$. Jestliže $e_{i,l,j} = 0$, pak platí z podmínky (2.25), že $e_{l,i,j} = 0$, a pak zase platí $x_{l,j} \leq s_{i,j}$.
Podmínka (2.26) je lineární a zaplatíme tím, že potřebujeme přidat podmínku (2.27). Obě podmínky pracují s konstantou G, kterou je nutné dobře odhadnout a obecně se říká, že je to dostatečné velké číslo. Používá se, tak aby za nějaké podmínky byla reálná proměnná omezená, ale tak velkým číslem, že by byla prakticky neomezená. Postup linearizace podmínek uvádí zdroj [33].

$$x_{i,j} - s_{l,j} \leq G(1 - e_{i,l,j}),\ i = 1, \ldots,\ n,\ l = 1, \ldots,\ n,\ i \neq l,\ j = 1, \ldots,\ m \tag{2.26}$$

$$x_{i,j} - s_{l,j} - (-G - 1)e_{i,l,j} \geq 1,\ i = 1, \ldots,\ n,\ l = 1, \ldots,\ n,\ i \neq l,\ j = 1, \ldots,\ m \tag{2.27}$$

Tyto podmínky nám určují posloupnosti procesů na každém stroji, ale pomocí nich dokážeme také zajistit, jestli se na stroji můžou entity předbíhat. Přesným určením některých hodnot $e_{i,l,j}$ tím dosáhneme.

Možnost předbíhání entit chceme dovolit pouze na paralelních strojích a na stroji M(p+2). Proto musíme pevně zadat podmínku (2.28) pro stroj M1. Tím jsme určili, že na stroji M1 se entity budou opracovávat v pořadí $1, 2, 3, \ldots,\ n$, ale na ostatních strojích

to může být jinak. Nerovnice (2.26) a (2.27) také počítají se všemi možnostmi výběru entit na paralelních strojích. Tím zajistíme, že jednoduše můžeme počítat i s více entitami na rozdíl od nelineárního modelu.

$$e_{i,l,1} = 1,\ i = 1, \ldots,\ n,\ l = 1, \ldots,\ n,\ i \leq l \tag{2.28}$$

Uvádíme shrnutí modelu se stále stejnou účelovou funkcí. V této kapitole jsme zjistili, že je nejvhodnější používat lineární model paralelní části výrobní linky. Za linearizování jsme zaplatili více nerovnicemi, avšak stále může být výpočet modelu rychlejší než nelineární verze. Další výhodou lineárního modelu je, že splňuje vlastnost předbíhání, se kterou lze snadno pracovat.

Tabulka 2.4: Model paralelní výrobní linky-lineární

Shrnutí modelu-paralelní výrobní linka-lineární verze	
$\min\ \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j}$	
$x_{i,j} - x_{i,1} \geq t_j d_{i,j}$	$i = 1, \ldots,\ n,\ j = 2, \ldots,\ (p+1)$
$x_{i,p+2} - x_{i,j} \geq t_{p+2} d_{i,j}$	$i = 1, \ldots,\ n,\ j = 2, \ldots,\ (p+1)$
$\sum_{j=1}^{p+1} d_{i,j} = 1$	$i = 1, \ldots,\ n$
$s_{i,j} + t_j d_{i,j} = x_{i,j}$	$i = 1, \ldots,\ n,\ j = 1, \ldots,\ m$
$d_{i,1} = 1$	$i = 1, \ldots,\ n$
$d_{i,p+2} = 1$	$i = 1, \ldots,\ n$
$e_{i,l,j} + e_{l,i,j} = 1$	$i = 1, \ldots,\ n,\ l = 1, \ldots,\ n,\ i \neq l,\ j = 1, \ldots,\ m$
$x_{i,j} - s_{l,j} \leq G(1 - e_{i,l,j})$	$i = 1, \ldots,\ n,\ l = 1, \ldots,\ n,\ i \neq l,\ j = 1, \ldots,\ m$
$x_{i,j} - s_{l,j} - (-G-1)e_{i,l,j} \geq 1$	$i = 1, \ldots,\ n,\ l = 1, \ldots,\ n,\ i \neq l,\ j = 1, \ldots,\ m$
$e_{i,l,1} = 1$	$i = 1, \ldots,\ n,\ l = 1, \ldots,\ n,\ i \leq l$
$x_{i,j},\ s_{i,j} \geq 0,\ d_{i,j},\ e_{i,l,j} \in \{0,1\}$	$i = 1, \ldots,\ n,\ l = 1, \ldots,\ n,\ i \neq l,\ j = 1, \ldots,\ m$

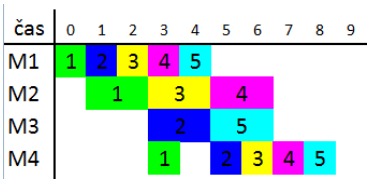
Tento model je lineární s celočíselnými a reálnými proměnnými MIP. Řešíme metodou větvení a mezí, kterou uvádí zdroj [18].

Příklad 4

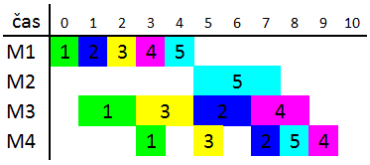
Zadání tohoto příkladu je stejné jako v předchozích příkladech s paralelní výrobní linkou podle obrázku 2.5. Po implementaci v programu GAMS, dostaneme výsledný Ganttův diagram 2.8. Zdrojový kód je uveden v příloze pod názvem *para_lin.gms*. Výsledek ukazuje, že 5 entit projde výrobní linkou opět za 9 časových dílků. Také vidíme podle rozřazení entit do paralelních strojů, že můžeme vyměnit procesy entit 2 a 3 a to stejné pro entity 4 a 5 bez změny výsledného času. Výměnou procesů entit 4 a 5 dostaneme logiku střídání. Model jsme sestavili s vlastností, že se mohou entity předbíhat. Předbíhání by se mělo projevit na stroji M4, avšak z důvodu zvolených procesních časů, se předbíhání neprojevilo. Proto procesní čas na stroji M2 prodloužíme a výpočtem dostaneme Ganttův diagram 2.9. Tady se již projevilo předbíhání, kde entita 5 předběhla entitu 4 na stroji M4 a 5 entit prošlo výrobní linkou za 10 časových dílků. Tabulka 2.5 uvádí zvolené procesní časy pro oba případy.

Tabulka 2.5: Tabulka zvolených procesních časů - příklad 4

Varianta/Procesní čas	t_1	t_2	t_3	t_4
Případ 1	1	2	2	1
Případ 2	1	3	2	1



Obrázek 2.8: Ganttův diagram lineárního modelu 1



Obrázek 2.9: Ganttův diagram lineárního modelu 2

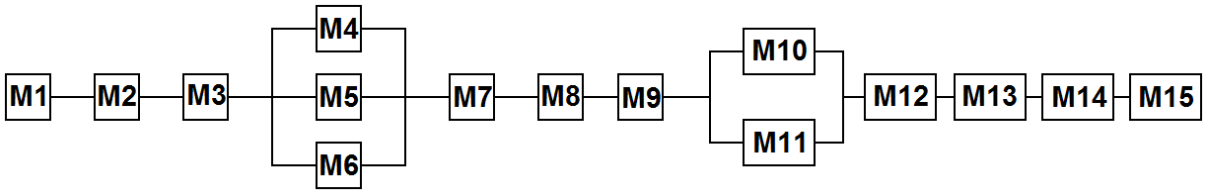
2.4 Obecná výrobní linka

Nyní budeme sestavovat model více kompaktněji, aby ho bylo snadné s ním pracovat pro velké množství entit a strojů. Tato kapitola se hlavně zabývá modelováním obecné výrobní linky, kterou jsme zavedli v kapitole 1. Postup modelování budeme provádět na obecné výrobní lince podle obrázku 2.10. Tato výrobní linka obsahuje 15 strojů a 2 části, ve kterých jsou 3 a 2 stroje zapojeny paralelně.

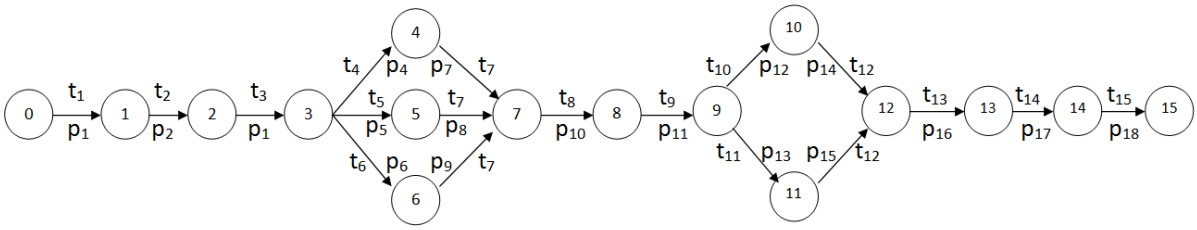
Můžeme se na výrobní linku podívat jako na graf, který je orientovaný, protože je možné s entitou jít pouze jedním směrem. Obrázek 2.11 ukazuje převedení obrázku 2.11 na orientovaný graf s ohodnocením hran. S tímto grafem budeme pracovat ve formě tzv. incidenční matice. Touto maticí bude zadána cesta každé entity.

Definice 11 *Incidenční maticí nazýváme matici typu $n \times c$, kde n je počet uzlů a c je počet hran. Prvky této matice jsou definovány podle (2.29).*

Incidenční matice zachycuje relace mezi uzly a hranami. Existuje několik typů těchto matic. Nejběžnější z nich jsou matice sousednosti uzlů grafu, matice sousednosti hran grafu a matice sousednosti uzlů a hran grafu. Definice těchto matic uvádí zdroj [3].



Obrázek 2.10: Obecná výrobní linka s 15-ti stroji



Obrázek 2.11: Orientovaný graf obecné výrobní linky s 15-ti stroji

Graf na obrázku 2.11 má uzly, které představují časovou posloupnost kroků zpracování entit. Hrany jsou ohodnoceny délkou procesu t_i. Můžeme si všimnout, že u paralelních řad se časy t_7 a t_{12} opakují. To bude mít za následek zavedení parametrů p_k, kde $k = 1, \ldots, 18$,

a které můžeme zavést do vektoru $\mathbf{p} = (p_1,\ p_2, \ldots,\ p_{18})$. Indexy k nemají nijak zvláštní smysl, jenom nám pomohou k sestavování modelu.

Zavedeme incidenční matici $\mathbf{B} = \{b_{i,k}\}$, která zachycuje vztahy mezi uzly a hranami. Matice bude mít sloupců stejně jako je hran v grafu, tedy v našem případě 18. Řádků bude mít 15, to je o jednu méně než je uzlů v grafu. Uzly v matici budou vystupovat od 1 do 15-ti. Uzel 0 je v grafu zaveden kvůli počáteční podmínce. Prvky incidenční matice jsou zavedeny podle předpisu (2.29).

$$b_{i,k} = \begin{cases} 1 & \text{pokud hrana } k \text{ je orientována do uzlu } i \\ -1 & \text{pokud hrana } k \text{ je orientována z uzlu } i \\ 0 & \text{pokud hrana } k \text{ nespojuje uzel } i \end{cases} \tag{2.29}$$

Jinak můžeme incidenční matici $\mathbf{B} = \{b_{i,k}\}$ chápat tak, že její sloupce jsou levé strany nerovnic, které popisují návaznosti mezi stroji jako v nerovnici (2.1). V našem příkladu bude matice (2.30) naší incidenční maticí $\mathbf{B}$.

$$\mathbf{B} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.30}$$

Tuto matici použijeme v soustavě nerovnic (2.31).

$$\mathbf{x}_i\mathbf{B} \geq \mathbf{p},\ \text{kde } i = 1, \ldots,\ n \tag{2.31}$$

Vektor $\mathbf{x}_i = (x_{i,1}, x_{i,2} \ldots,\ x_{i,15})$ obsahuje neznámé proměnné a opět mají význam času dokončení entity i na stroji j. V modelu vystupuje n soustav nerovnic, to je jedna soustava pro jednu entitu. Tím jsme popsali cestu každé entity a model se chová stejně jako v kapitole 2.1. V našem modelu máme paralelní stroje. To, jak již víme z předchozí kapitoly, znamená, že entita může projít pouze jedním z těchto strojů. Takže musíme opět zavést binární proměnné $d_{i,k}$ zavedeny podle předpisu (2.5). Přidáním těchto proměnných do soustavy (2.31), dostaneme správnou soustavu (2.32). Znakem * značíme násobení vektorů po složkách. To znamená $u * v = (u_1v_1,\ u_2v_2, \ldots,\ u_kv_k)$.

$$\mathbf{x}_i\mathbf{B} \geq \mathbf{p} * \mathbf{d}_i,\ \text{kde } i = 1, \ldots,\ n. \tag{2.32}$$

Proměnné $d_{i,k}$ použijeme jako vektor $\mathbf{d}_i = (d_{i,1}, d_{i,2} \ldots, d_{i,18})$. Dále je nutné, aby platily podmínky (2.33) až (2.39).

$$d_{i,4} + d_{i,5} + d_{i,6} = 1,\ i = 1, \ldots,\ n \tag{2.33}$$

$$d_{i,12} + d_{i,13} = 1,\ i = 1, \ldots,\ n \tag{2.34}$$

$$d_{i,4} = d_{i,7},\ i = 1, \ldots,\ n \tag{2.35}$$

$$d_{i,5} = d_{i,8},\ i = 1, \ldots,\ n \tag{2.36}$$

$$d_{i,6} = d_{i,9},\ i = 1, \ldots,\ n \tag{2.37}$$

$$d_{i,12} = d_{i,14},\ i = 1, \ldots,\ n \tag{2.38}$$

$$d_{i,13} = d_{i,15},\ i = 1, \ldots,\ n \tag{2.39}$$

Podmínky (2.33) až (2.34) nám zajistí, že bude platit pouze jeden sloupec matice $\mathbf{B}$ ze sloupců týkajících se paralelních strojů. Podmínky (2.35) až (2.39) nám zajišťují, že když entita vstoupí do jednoho ze strojů, tak že také z toho stroje půjde do dalšího. Protože teď máme $d_{i,k}$ použité na všechny sloupce matice $\mathbf{B}$, musíme pevně zadat $d_{i,k}$, které se týkají sériových strojů.

$$d_{i,k} = 1,\ i = 1, \ldots,\ n,\ k = 1, \ldots,\ 18,\ k \neq 4, 5, 6, 7, 8, 9, 12, 13, 14, 15 \tag{2.40}$$

Teď se budeme zabývat tím, aby procesy na jednotlivých strojích měly nějakou posloupnost a nemohly se překrývat. Tím máme na mysli zobecnění nerovnice (2.2). Narozdíl od minulého modelu, bychom chtěli dovolit předbíhání entit, které je zapotřebí u paralelních strojů. Musíme tedy popsat podmínkami, aby platilo, že například entita 5 předchází na stroji j entitu 2, ale naopak to platit nesmí. Musíme proto zavést nové proměnné. Nejdříve proměnnou $s_{i,j}$, kde $i = 1, \ldots,\ n,\ j = 1, \ldots,\ m$. Tyto proměnné představují čas začátku procesu entity i na stroji j a budou platit podmínky (2.41) pro sériové stroje a (2.42) pro paralelní stroje.

$$s_{i,j} + t_j = x_{i,j},\ i = 1, \ldots,\ n,\ j = 1, \ldots,\ m \text{ a } j \neq 4, 5, 6, 10, 11 \tag{2.41}$$

$$s_{i,j} + t_j d_{i,k} = x_{i,j},\ i = 1, \ldots,\ n,\ (j,k) = (4,4), (5,5), (6,6), (10,12), (11,13) \tag{2.42}$$

Tyto podmínky říkají, že pokud k času začátku procesu přičteme čas trvání procesu, tak dostaneme čas konce procesu. Toto platí pro to, kdy jsou proměnné $d_{i,k}$ rovny jedné. Pokud jsou nulové, tak to bude znamenat, že čas začátku procesu se bude rovnat času konce procesu. To má pouze smysl pro účely modelu, aby jsme zadefinovali, čemu se budou rovnat proměnné $s_{i,j}$ a $x_{i,j}$ pro $d_{i,j}$ rovné nule. Tedy pro ty proměnné, které přísluší paralelním strojům, kterými entita neprojde.

Další proměnné, které zadefinujeme jsou opět binární $e_{i,l,j}$, kde $i = 1, \ldots, n$, $l = 1, \ldots, n,\ i \neq l,\ j = 1, \ldots, m$. Tyto proměnné jsme použili v podkapitole 2.3 a jsou definovány stejně podle (2.24). K těmto proměnným zadáme stejné podmínky (2.25) s lineárními podmínkami (2.26) a (2.27).

Tyto podmínky a nerovnice nám určují posloupnosti procesů na každém stroji, ale pomocí nich dokážeme také zajistit, jestli se na stroji můžou entity předbíhat. Přesným určením některých hodnot $e_{i,l,j}$ toho dosáhneme.

Jestliže v prvním stroji označíme entity od 1 do n, tak chceme aby v tomto pořadí byly zpracovány až do stroje 3.

$$e_{i,l,j} = 1,\ i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \leq l,\ j = 1, \ldots, 3 \tag{2.43}$$

V paralelních strojích 4, 5, 6 požadujeme, aby se entity seřazovaly libovolně a mohly se předběhnout. Tady $e_{i,j,l}$ určovat nebudeme a to ani pro stroj 7. Po tom co se entity seřadí ve stroji 7, tak zase v tomto pořadí musí pokračovat přes další sériové stroje.

$$e_{i,l,7} = e_{i,l,j},\ i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 8, 9 \tag{2.44}$$

Analogicky e_{ilj} určíme pro další paralelní část.

$$e_{i,l,12} = e_{i,l,j},\ i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 13, 14, 15 \tag{2.45}$$

Můžeme si ještě doplnit podmínku na plnění prvního stroje. Do něho mohou být generovány entity po různých časových úsecích nebo náhodně. My se pro zjednodušení zaměříme na situaci, kdy se každá entita vygeneruje po stejném časovém úseku. Zavedeme vektor poc_i, jehož složky jsou minimální časy, kdy entita i může být opracovávána na prvním stroji. Tento vektor bude pro naše zjednodušení ekvidistantní a bude pro něho platit podmínka (2.46).

$$s_{i,1} \geq poc_i,\ i = 1, \ldots, n \tag{2.46}$$

Podmínky modelu jsou v tuto chvíli hotovy a můžeme opět použít účelovou funkci (2.3). Tentokrát se jedná o úlohu lineárního programování s celočíselnými a reálnými proměnnými MIP. Opět se tyto úlohy mohou být řešeny metodou větvení a mezí.

Tabulka 2.6: Model obecné výrobní linky

Shrnutí modelu-obecná výrobní linka	
$\min \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j}$	
$s_{i,1} \geq poc_i$	$i = 1, \ldots, n$
$\mathbf{Bx}_i \geq \mathbf{pd}_i$	$i = 1, \ldots, n$
$d_{i,4} + d_{i,5} + d_{i,6} = 1$	$i = 1, \ldots, n$
$d_{i,12} + d_{i,13} = 1$	$i = 1, \ldots, n$
$d_{i,4} = d_{i,7}$	$i = 1, \ldots, n$
$d_{i,5} = d_{i,8}$	$i = 1, \ldots, n$
$d_{i,6} = d_{i,9}$	$i = 1, \ldots, n$
$d_{i,12} = d_{i,14}$	$i = 1, \ldots, n$
$d_{i,13} = d_{i,15}$	$i = 1, \ldots, n$
$d_{i,k} = 1$	$i = 1, \ldots, n,\ k = 1, \ldots, 18,$ $k \neq 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 12,\ 13,\ 14,\ 15$
$s_{i,j} + t_j = x_{i,j}$	$i = 1, \ldots, n,\ j = 1, \ldots, m$ a $j \neq 4,\ 5,\ 6,\ 10,\ 11$
$s_{i,j} + t_j d_{i,k} = x_{i,j}$	$i = 1, \ldots, n,$ $(j,\ k) = (4,\ 4),\ (5,\ 5),\ (6,\ 6),\ (10,\ 12),\ (11,\ 13)$
$e_{i,l,j} + e_{l,i,j} = 1$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 1, \ldots, m$
$e_{i,l,j} = 1$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \leq l,\ j = 1, \ldots, 3$
$e_{i,l,7} = e_{i,l,j}$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 8,\ 9$
$e_{i,l,12} = e_{i,l,j}$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 13,\ 14,\ 15$
$x_{i,j} - s_{l,j} \leq G(1 - e_{i,l,j})$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 1, \ldots, m$
$x_{i,j} - s_{l,j} - (-G-1)e_{i,l,j} \geq 1$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 1, \ldots, m$
$x_{i,j},\ s_{i,j} \geq 0,$ $d_{i,k},\ e_{i,l,j} \in \{0, 1\}$	$i = 1, \ldots, n,\ k = 1, \ldots, 18,\ j = 1, \ldots, m$

Příklad 5

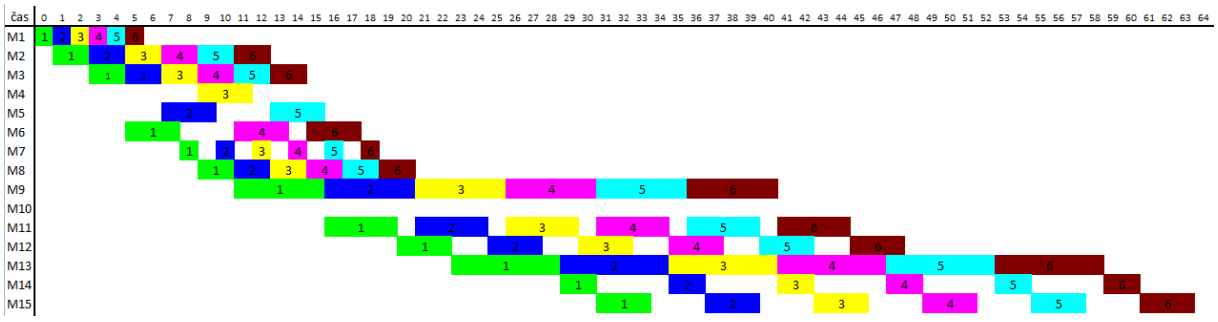
Zadání příkladu je podle obrázku (2.10). Tabulka představuje hodnoty přiřazené parametrům t_j a p_k, které byly použity v tomto příkladu. Implementování bylo provedeno v programu GAMS a zdrojový kód najdeme v příloze pod názvem *obecnalinka.gms* .

Tabulka 2.7: Tabulka zvolených procesních časů - příklad 5

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_7	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{12}	t_{13}	t_{14}	t_{15}
p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	p_{17}	p_{18}
1	2	2	3	3	3	1	1	1	2	5	4	4	3	3	6	2	3

Na obrázku 2.12 vidíme Ganttův digram pro náš příklad. I když jsme zvolili, aby se entity přes paralelní stroje předbíhaly, tak se stále opracovávají ve stejném pořadí. To je způsobeno charakterem této linky a její volbou délek procesů na strojích. Z tohoto výsledku můžeme usoudit, že použití nerovnic (2.57) a (2.58) je zbytečné, ale pouze u sériových strojů. U nich bychom nerovnice nahradili jednoduchými nerovnicemi (2.2). Práce s tímto modelem je velice individuální, ale je možné určit postup a princip sestavení modelu. Z každým jiným příkladem je však potřeba vytvořit originální model, hlavně kvůli paralelním strojům a jejich složitosti.

Další výsledky, které můžeme zjistit z Ganttova diagramu 2.12 jsou, že výrobní linkou projde 6 entit za 64 časových dílků. Dále vidíme, že u paralelních stroj M4, M5 a M6 se entity střídají až na entitu 6. Výsledek by byl však nezměněn, kdyby entita 6 byla opracována ve stroji M4. Takže můžeme usoudit, že optimálním rozřazováním pro tyto stroje je logika střídání a rovnou s ní počítat. U paralelních strojů M10 a M11 se ukázalo, že jeden paralelní stroj je zbytečný. Všechny entity prošly pouze jedním strojem a rozřazením mezi oba stroje by k zlepšení v tomto příkladu nedošlo.



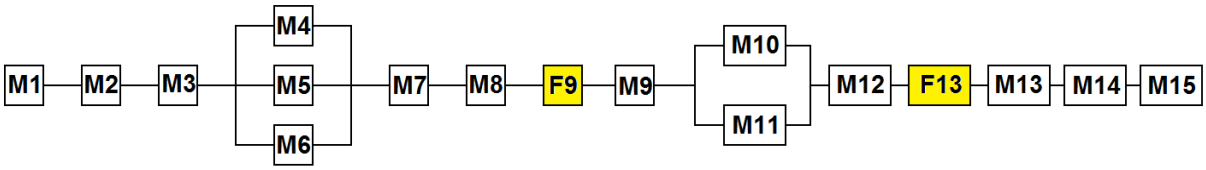
Obrázek 2.12: Ganttův diagram pro m=15 a n=6

2.5 Model s frontami

V našem modelu budeme chtít zahrnout fronty, tak abychom s nimi mohli pracovat. Uděláme to jednoduše tak, že fronta bude také stroj, ale s jinými vlastnostmi. Vztahy (2.1), které popisují cestu entity přes stroje budou stále platit. Vztahy (2.2), které popisují uspořádání procesů entit na frontu nepoužijeme. To bude znamenat, že na stroji, který bude představovat frontu, bude moci být více entit v jednom čase. Délku procesu entity t_j příslušné frontě nastavíme na nulu, ale není důležité na jakou hodnotu parametr nastavíme, protože ho nepoužijeme. Místo tohoto parametru zavedeme proměnnou $f_{i,r}$, kde $r = 1, \ldots, \ q$. Parametrem q značíme počet front. Hodnota $f_{i,r}$ představuje pro entitu i dobu strávenou ve frontě r. Těmito úpravami docílíme pravidla FIFO, pokud bychom chtěli jiné pravidlo pro fronty, musely by se omezení přizpůsobit. Další úpravy modelu s frontami ukážeme na příkladu, protože změny nejsou obecné ale individuální na umístění front. Shrnutí modelu podle obrázku 2.13 a uvedeme na konci příkladu.

Příklad 6

V tomto příkladu se budeme zabývat stejnou situací jako v předchozí kapitole. Do výrobní linky umístíme dvě fronty F9 a F14 podle obrázku (2.13). Postupnými kroky budeme původní model upravovat.



Obrázek 2.13: Obecná výrobní linka se dvěma přidanými frontami

Vztah (2.32) nahradíme vztahem (2.47). Podmínkou $k \neq 11,\ 17$ vynecháme 2 sloupce incidenční matice $\mathbf{B}$, které se týkají fronty.

$$\sum_{j=1}^{m} x_{i,j} b_{i,k} \geq p_k d_{i,k},\ i = 1, \ldots, \ n,\ k = 1, \ldots, \ 20,\ k \neq 11,\ 17 \tag{2.47}$$

Odstranili jsme určitá omezení, ale ty nahradíme vztahy (2.48) a (2.49). Docílili jsme toho, že jsme parametry t_j pro $j = 11, 17$ nahradili proměnnými $f_{i,r}$ pro $r = 1, 2$.

$$x_{i,9} - x_{i,8} \geq f_{i,1},\ i = 1, \ldots, \ n \tag{2.48}$$

$$x_{i,14} - x_{i,13} \geq f_{i,2},\ i = 1, \ldots, \ n \tag{2.49}$$

Rovnice (2.41) musíme zadat také speciálně pro fronty. Pro $j = 9,\ 14$ nebudou tyto rovnice platit a nahradíme je vztahy (2.50) a (2.51). Vztahy (2.52) a (2.53) definují dobu entity i ve frontě r $f_{i,r}$, tak že je to čas mezi vstupem entity do stroje za frontou a výstupem ze stroje před frontou.

$$s_{i,9} + f_{i,1} = x_{i,9},\ i = 1, \ldots,\ n \tag{2.50}$$

$$s_{i,14} + f_{i,2} = x_{i,14},\ i = 1, \ldots,\ n \tag{2.51}$$

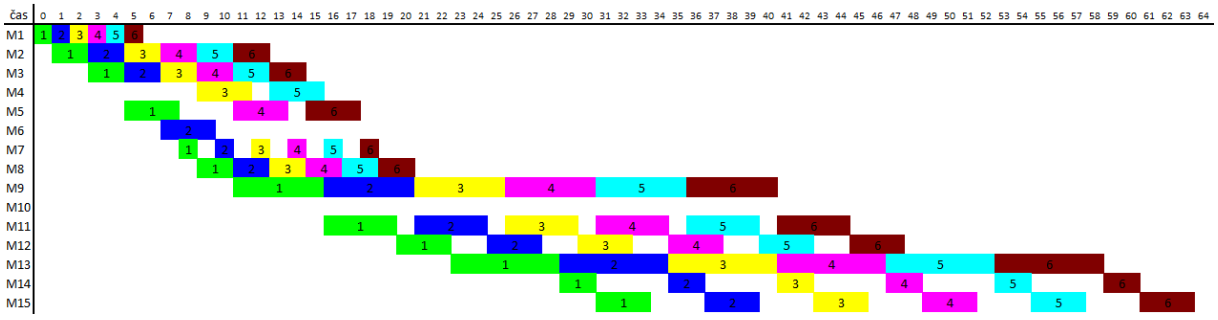
$$f_{i,1} = s_{i,10} - x_{i,8},\ i = 1, \ldots,\ n \tag{2.52}$$

$$f_{i2} = s_{i,15} - x_{i,13},\ i = 1, \ldots,\ n \tag{2.53}$$

Tím jsme provedli nejdůležitější úpravy modelu pro tento příklad. Dále musíme pouze přeindexovat některé proměnné ve vztazích, kvůli přidání sloupců front do incidenční matice a jejích posunutí. Tento příklad jsme vyřešili v programu GAMS a dostali jsme Ganttův diagram 2.14. To kolik bylo entit ve frontách F9 a F14 udává obrázek 2.15. Zdrojový kód najdeme v příloze pod názvem *fronty.gms*. Procesní časy pro tento příklad uvádí tabulka 2.8, kde jsou vynechány časy t_9 a t_{14}, protože jsou nulové a v modelu se s nimi nepočítá, jak již bylo zmíněno.

Tabulka 2.8: Tabulka zvolených procesních časů - příklad 6

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_7	t_7	t_8	t_{10}	t_{11}	t_{12}	t_{12}	t_{13}	t_{15}	t_{16}	t_{17}
p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{12}	p_{13}	p_{14}	p_{15}	p_{16}	p_{18}	p_{19}	p_{20}
1	2	2	3	3	3	1	1	1	2	5	4	4	3	3	6	2	3



Obrázek 2.14: Ganttův diagram obecné výrobní linky s frontami F9 a F13

Z výsledku vidíme, že 6 entit výrobní linkou projde za 64 časových dílků. Tvoření front se nejvíce projevilo ve frontě F9, kde čekaly až 3 entity. Ve frontě F13 entity také čekaly ve frontě, ale vždy maximálně jedna.

Tabulka 2.9: Model obecné výrobní linky s frontami

Shrnutí modelu-obecná výrobní linka s frontami F9 a F13	
$\min \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j}$	
$s_{i,1} \geq poc_i$	$i = 1, \ldots, n$
$\sum_{j=1}^{m} x_{i,j} b_{i,k} \geq p_k d_{i,k}$	$i = 1, \ldots, n,\ k = 1, \ldots, 20,\ k \neq 11,\ 17$
$d_{i,4} + d_{i,5} + d_{i,6} = 1$	$i = 1, \ldots, n$
$d_{i,13} + d_{i,14} = 1$	$i = 1, \ldots, n$
$d_{i,4} = d_{i,7}$	$i = 1, \ldots, n$
$d_{i,5} = d_{i,8}$	$i = 1, \ldots, n$
$d_{i,6} = d_{i,9}$	$i = 1, \ldots, n$
$d_{i,13} = d_{i,15}$	$i = 1, \ldots, n$
$d_{i,14} = d_{i,16}$	$i = 1, \ldots, n$
$d_{i,k} = 1$	$i = 1, \ldots, n,\ k = 1, \ldots, 20,$ $k \neq 4,\ 5,\ 6,\ 7,\ 8,\ 9,\ 13,\ 14,\ 15,\ 16$
$s_{i,j} + t_j = x_{i,j}$	$i = 1, \ldots, n,\ j = 1, \ldots, m,$ $j \neq 4,\ 5,\ 6,\ 9,\ 11,\ 12,\ 14$
$s_{i,j} + t_j d_{i,k} = x_{i,j}$	$i = 1, \ldots, n,$ $(j,\ k) = (4,\ 4),\ (5,\ 5),\ (6,\ 6),\ (11,\ 13),\ (12,\ 14)$
$e_{i,l,j} + e_{l,i,j} = 1$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 1, \ldots, m$
$e_{i,l,j} = 1$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \leq l,\ j = 1, \ldots, 3$
$e_{i,l,7} = e_{i,l,j}$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 8,\ 10$
$e_{i,l,13} = e_{i,l,j}$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 15,\ 16,\ 17$
$x_{i,j} - s_{l,j} \leq G(1 - e_{i,l,j})$	$i = 1, \ldots, n,\ l = 1, \ldots, n,$ $i \neq l,\ j = 1, \ldots, m,\ j \neq 9,\ 14$
$x_{i,j} - s_{l,j} - (-G - 1)e_{i,l,j} \geq 1$	$i = 1, \ldots, n,\ l = 1, \ldots, n,$ $i \neq l,\ j = 1, \ldots, m,\ j \neq 9,\ 14$
$x_{i,j},\ s_{i,j},\ f_{i,r} \geq 0,$ $d_{i,k},\ e_{i,l,j} \in \{0, 1\}$	$i = 1, \ldots, n,\ k = 1, \ldots, 20,\ j = 1, \ldots, m,\ r = 1,\ 2$
$x_{i,9} - x_{i,8} \geq f_{i,1}$	$i = 1, \ldots, n$
$x_{i,14} - x_{i,13} \geq f_{i,2}$	$i = 1, \ldots, n$
$s_{i,9} + f_{i,1} = x_{i,9}$	$i = 1, \ldots, n$
$f_{i,1} = s_{i,10} - x_{i,8}$	$i = 1, \ldots, n$
$s_{i,14} + f_{i,2} = x_{i,14}$	$i = 1, \ldots, n$
$f_{i,2} = s_{i,15} - x_{i,13}$	$i = 1, \ldots, n$

čas	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
F9	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2	1	2	2	3	3	3	3	3	3	3	2	2	2	2	2	1	1
F13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

čas	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
F9	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F13	1	1	0	0	0	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Obrázek 2.15: Počet entit ve frontě F9 a F13 v daném čase

2.6 Model s údržbami

Další oblastí, kterou se můžeme ve výrobní lince zabývat, jsou údržby. V praxi není obvyklé, aby výrobní linka byla v provozu bez přerušování. Často se stroje porouchávají z různých důvodů. Nebo je nutné naplánovat pravidelné údržby během provozu. Může být zadáno několik podmínek. Například podmínka na délku údržby na každém stroji a kolik jich může být prováděno naráz. Cílem by mělo být naplánovat časy údržeb, aby co nejméně omezovaly plán výroby. To znamená, aby bylo vyrobeno co nejvíce výrobků.

Náš model z kapitoly 2.4 upravíme jednoduše tak, že budeme považovat jednu entitu za údržbu. Potom upravíme pár vztahů. Nejprve dovolíme, aby se entita mohla řadit libovolně v čase. To prakticky znamená, že proměnné $e_{i,l,j}$, pro i nebo l týkající se procesu držby, necháme volné pro každé j. Volné znamená, že jim nebudeme přiřazovat pevnou hodnotu. Dále musíme zavést parametr u_j, kde $j = 1, \dots, m$. Ten bude určovat čas začátku údržby na stroji j.

Příklad 7

Jako entitu údržby zvolíme například entitu 6. Hodnotám $t_{6,j}$ přiřadíme časy trvání údržby na stroji j. Procesní časy použijeme z tabulky 2.7. Parametr u_j dáme s proměnnou $s_{i,j}$ do vztahu (2.54).

$$s_{i,j} = u_j,\ j = 1, \dots,\ m,\ i = 6 \tag{2.54}$$

V tomto okamžiku si můžeme pevně volit začátky údržeb na všech strojích. Pokud nechceme na některém stroji konat údržbu, stačí zvolit dobu trvání $t_{6,j} = 0$. My bychom, ale rádi začátky údržeb neměli pevně zvolené, ale proměnné. Potom bychom hledali optimální čas údržeb a rovnici (2.54) nezahrnovali nebo u_j zvolili jako proměnnou.

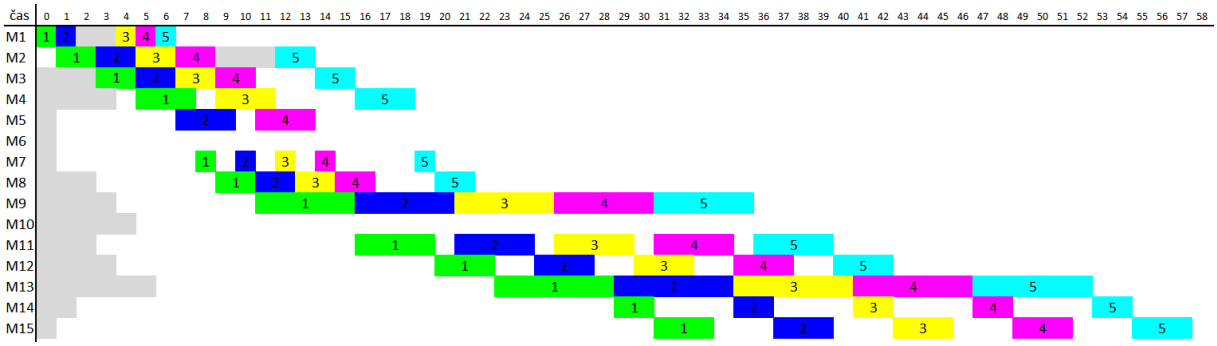
Po spočítání a použití hodnot z tabulky 2.11, na obrázku 2.17 vidíme časy optimálních údržeb, které jsou označeny šedou barvou. Jak jsme mohli předpokládat, výsledek ukázal, že máme údržby na většině strojích provádět před zpracováním první entity. Avšak tento výsledek není nejspíš použitelný, protože budeme chtít vědět, kdy stroje zastavit za delší čas od začátku provozu. Výsledek, ale aspoň posloužil ke kontrole. S těmito údržbami výrobní linkou prošlo 5 entit za čas 58 časových dílků. Měli bychom tedy určit jakýsi hraniční čas, kdy je každý stroj linky v provozu. Tento čas by měl také určovat minimální

Tabulka 2.10: Model obecné výrobní linky s údržbami

Shrnutí modelu-obecná výrobní linka s údržbami	
$\min \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j}$	
$s_{i,1} \geq poc_i$	$i = 1, \ldots, n$
$\mathbf{Bx}_i \geq \mathbf{pd}_i$	$i = 1, \ldots, n$
$d_{i,4} + d_{i,5} + d_{i,6} = 1$	$i = 1, \ldots, n$
$d_{i,12} + d_{i,13} = 1$	$i = 1, \ldots, n$
$d_{i,4} = d_{i,7}$	$i = 1, \ldots, n$
$d_{i,5} = d_{i,8}$	$i = 1, \ldots, n$
$d_{i,6} = d_{i,9}$	$i = 1, \ldots, n$
$d_{i,12} = d_{i,14}$	$i = 1, \ldots, n$
$d_{i,13} = d_{i,15}$	$i = 1, \ldots, n$
$d_{i,k} = 1$	$i = 1, \ldots, n,\ k = 1, \ldots, 18,$ $k \neq 4, 5, 6, 7, 8, 9, 12, 13, 14, 15$
$d_{6,4} = 1$	
$d_{6,5} = 1$	
$d_{6,6} = 1$	
$d_{6,12} = 1$	
$d_{6,13} = 1$	
$s_{i,j} + t_j = x_{i,j}$	$i = 1, \ldots, n,\ j = 1, \ldots, m$ a $j \neq 4, 5, 6, 10, 11$
$s_{i,j} + t_j d_{i,k} = x_{i,j}$	$i = 1, \ldots, n,$ $(j,\ k) = (4,\ 4),\ (5,\ 5),\ (6,\ 6),\ (10,\ 12),\ (11,\ 13)$
$e_{i,l,j} + e_{l,i,j} = 1$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 1, \ldots, m$
$e_{i,l,j} = 1$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \leq l,\ j = 1, \ldots, 3$
$e_{i,l,7} = e_{i,l,j}$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 8,\ 9$
$e_{i,l,12} = e_{i,l,j}$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 13,\ 14,\ 15$
$x_{i,j} - s_{l,j} \leq G(1 - e_{i,l,j})$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 1, \ldots, m$
$x_{i,j} - s_{l,j} - (-G-1)e_{i,l,j} \geq 1$	$i = 1, \ldots, n,\ l = 1, \ldots, n,\ i \neq l,\ j = 1, \ldots, m$
$s_{i,j} \geq u_j$	$j = 1, \ldots, m,\ i = 6$
$x_{6,j} - s_{6,j'} \leq G(1 - a_{j,j'})$	$j = 1, \ldots, m,\ j' = 1, \ldots, m,\ j \neq j'$
$x_{6,j} - s_{6,j'} - (-G-1)a_{j,j'} \geq 1$	$j = 1, \ldots, m,\ j' = 1, \ldots, m,\ j \neq j'$
$a_{j,j'} + a_{j',j} = 1$	$j = 1, \ldots, m,\ j' = 1, \ldots, m,\ j \neq j'$
$x_{i,j},\ s_{i,j} \geq 0,$ $d_{i,j},\ a_{j,j'},\ e_{i,l,j} \in \{0, 1\}$	$i = 1, \ldots, n,\ k = 1, \ldots, 18,\ j = 1, \ldots, m,$ $j' = 1, \ldots, m,\ j \neq j'$

Tabulka 2.11: Tabulka zvolených délek údržeb

$t_{6,1}$	$t_{6,2}$	$t_{6,3}$	$t_{6,4}$	$t_{6,5}$	$t_{6,6}$	$t_{6,7}$	$t_{6,8}$	$t_{6,9}$	$t_{6,10}$	$t_{6,11}$	$t_{6,12}$	$t_{6,13}$	$t_{6,14}$	$t_{6,15}$
1	3	5	8	10	7	9	11	16	25	20	23	27	28	31



Obrázek 2.16: Ganttův diagram s optimálními údržbami 1

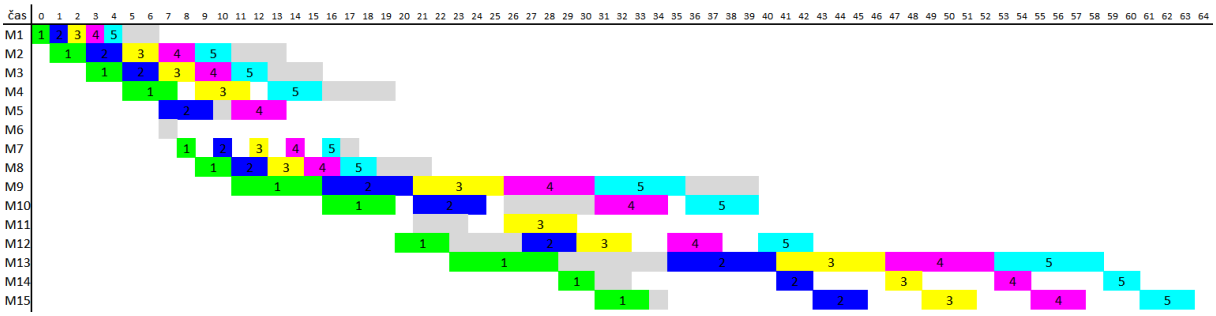
požadovanou dobu provádění údržeb dle zvoleného kritéria. Tento čas bychom měli zavést nejlépe pro každý stroj jiný. Takže můžeme použit vektor u_j a přidat podmínku (2.55).

$$s_{i,j} \geq u_j, \;\; j = 1, \ldots, \; m, \;\; i = 6 \tag{2.55}$$

Tabulka 2.12: Tabulka zvolených hodnot u_j

u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}	u_{12}	u_{13}	u_{14}	u_{15}
1	3	5	8	10	7	9	11	16	25	20	23	27	28	31

Po použití hodnot podle tabulky 2.12, dostaneme výsledek, který znázorňuje obrázek 2.17. Tady vidíme, že se údržby objevují již v provozu strojů, a že výrobní linkou prošlo 5 entit za 64 časových dílků. Obdobně by jsme zavedli maximální časy údržeb každého stroje. Jiné podmínky závisí na požadavcích firmy a není těžké je do modelu přidat. Dále musíme poznamenat, že by byly výsledky zajímavější, kdybychom pracovali s více entitami. Pro tyto ukázkové příklady, však používáme 5 entit, abychom neztráceli čas s dlouhými výpočtovými časy.



Obrázek 2.17: Ganttův diagram s optimálními údržbami 2

Příklad 8

Na obrázku 2.17 si u některých strojů můžeme všimnout, že časy údržeb jsou ve stejných chvílích. Bylo by dobré ještě dodat omezení na to, kolik může být zároveň prováděno údržeb. To je nezbytné, protože nemusíme mít dostatek dělníků na opravy ve stejný čas. Proto musíme doplnit podmínku na maximální počet údržeb prováděných v jednom čase. K tomu nám poslouží podmínky (2.25), (2.26) a (2.27). Zavedeme binární proměnné $a_{j,j'}$ podle předpisu (2.56), kde $j = 1, \ldots, m$, $j' = 1, \ldots, m$, $j \neq j'$.

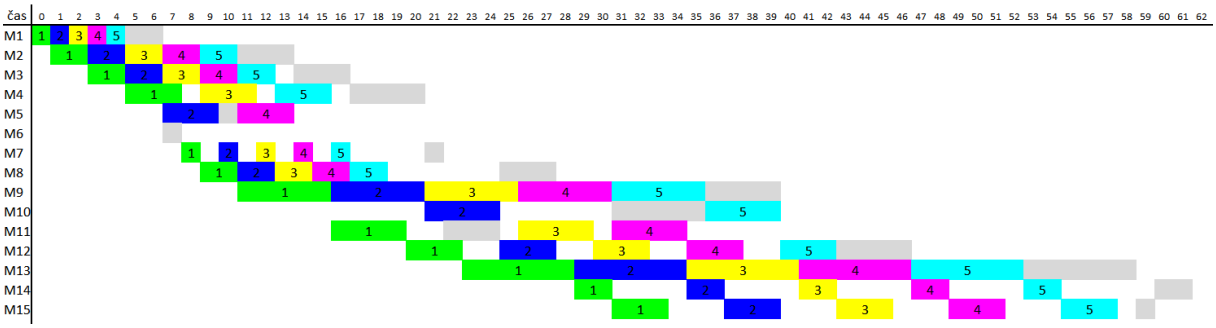
$$a_{j,j'} = \begin{cases} 1 & \text{pokud údržba na stroji } j \text{ je prováděna před údržbou na stroji } j' \\ 0 & \text{pokud údržba na stroji } j' \text{ je prováděna před údržbou na stroji } j \end{cases} \tag{2.56}$$

Potom stačí přidat podmínky (2.57),(2.58) a logickou podmínku (2.59). Jejich význam známe z kapitoly 2.4. Implementovaný model najdeme v příloze pod názvem *udrzby.gms* .

$$x_{6,j} - s_{6,j'} \leq G(1 - a_{j,j'}) \tag{2.57}$$

$$x_{6,j} - s_{6,j'} - (-G - 1)a_{j,j'} \geq 1 \tag{2.58}$$

$$a_{j,j'} + a_{j',j} = 1, \; j = 1, \ldots, \; m, \; j' = 1, \ldots, \; m, \; j \neq j' \tag{2.59}$$



Obrázek 2.18: Ganttův diagram s optimálními údržbami 3

Zajistili jsme uspořádání údržeb, tedy jak se mají předcházet a také, že se nemají překrývat. Výsledný Ganttův diagram vidíme na obrázku 2.18. Opět za 58 časových dílků projde výrobní linkou 5 entit. To by mohla být postačující podmínka pro úlohu, že máme pouze jednoho údržbáře. Také to můžeme brát z toho pohledu, že je lepší když jsou údržby prováděny po sobě a nehromadí se naráz. Kdybychom chtěli docílit podmínky, že připustíme nějaký maximální počet provádění údržeb v jednom čase, který je větší než jedna, je potom úloha takto těžce řešitelná. Nelze lehce matematicky tuto podmínku uchopit pro tolik strojů a tak rozmanitou linku. Dal by se problém řešit pro konkrétní zadání u požadovaných strojů. Pokud bychom se zabývali počítačovým programováním tohoto problému, šlo by nejspíš s naším požadavkem pracovat. Další podmínkou na údržby, která by mohla být užitečná je ta, že chceme dělat více údržeb na jednom stroji. To se může hodit i z hlediska toho, že by jsme chtěli údržby provádět opakovaně po nějakých okamžicích, například každých 8 hodin. Není těžké tento požadavek splnit, stačí zopakovat postup pro zavedení údržeb do modelu na začátku této kapitoly. Vezmeme tedy další entity, které prohlásíme za proces údržby a zvolíme požadované podmínky z této kapitoly. Vektor u_j rozšíříme na matici $u_{r,j}$, jejíž řádku budou vektory u_j pro r-tou údržbu. Celý uvedený přístup předpokládá, že údržby probíhají v dobách, kdy jsou stroje prázdné.

Kapitola 3

Model výrobní linky - simulační přístup

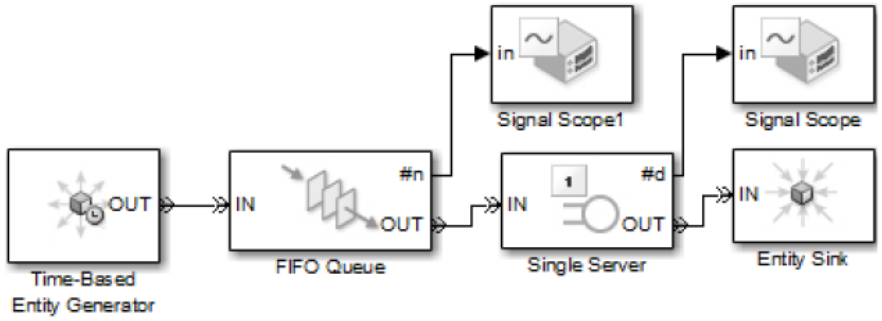
V kapitole 2 jsme vytvářeli matematické modely výrobních linek. Zabývali jsem se matematickým popisem chování a vlastností výrobních linek. Následně jsme hledali optimální doby údržeb. Analyzování výrobních linek můžeme ještě řešit simulačním přístupem. A tímto přístupem se budeme zabývat v této kapitole.

Známou disciplínou, která zkoumá obslužné systémy, je teorie front [19] neboli teorie hromadné obsluhy. Touto disciplínou se zabýval matematik D.G. Kendall v 50. letech minulého století. Jedním z kritérií, jak tuto teorii rozdělit, je podle typu řešení, a to na analytický přístup nebo simulační. Simulační přístup využívá známých parametrů a výsledkem je simulace pomocí vhodného softwaru. Simulovaný model musí věrně splňovat rysy zadané situace. Naší situací tedy bude výrobní linka. Zdroj [10] pojednává o analytickém přístupu k teorii front.

Existuje několik softwarů pro řešení simulačního přístupu. My použijeme software SimEvents, který je součástí programu Matlab-Simulink. SimEvents poskytuje diskrétní simulátor a komponenty pro analyzování diskrétně řízených systémů. Pomáhá optimalizovat různé charakteristiky jako například fronty. Sestavení simulace probíhá spojováním bloků jako jsou fronty, servery, rozbočovače a jiné. S tímto softwarem lze studovat odchod entit ze strojů, počet entit ve frontě, čekací doby atd. Program také dovoluje použití speciálních bloků, které slouží k přidávání matlabovských kódů. Tímto je nabídnuta velká volnost v použití programu SimEvents. K programu existuje základní příručka [28] a článek [9] pojednává o jeho užitečných vlastnostech pro konstrukci diskrétních systémů.

Na obrázku 3.1 vidíme základní zapojené prvky programu SimEvents. Tato ukázka představuje simulaci výrobní linky s jedním strojem. Každá simulace není tvořena pouze servery, které znázorňují stroje, ale také potřebnými bloky pro generování entit, pro výstup entit, pro vizualizaci statistik a pro správu front. Tyto bloky krátce popíšeme.

Blok *Time-Based Entity generator* se používá jako vstupní blok a slouží pro generování entit v závislosti na čase. Obrázek 3.2 zobrazuje nastavení tohoto bloku. Generování entit

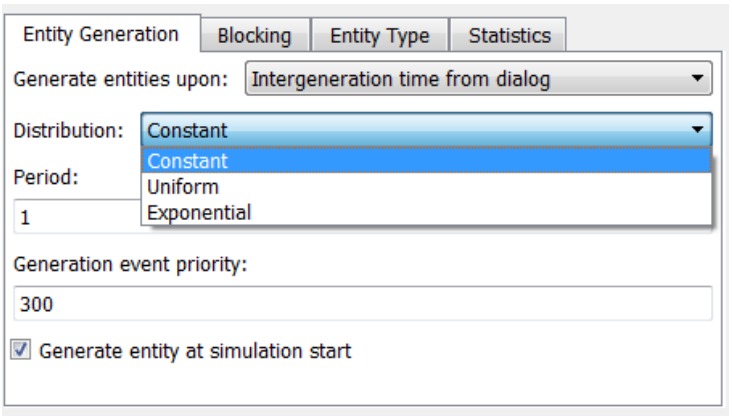


Obrázek 3.1: Ukázka zapojení základních prvků

lze nastavit na konstantní, uniformní nebo exponenciální distribuci. Popis těchto distribucí najdeme v tabulce 3.1. Každá tato možnost vyžaduje vyplnění určitých parametrů. Generování entit lze také ovládat z vstupního portu tohoto bloku, který může být například napojen na generátor signálu.

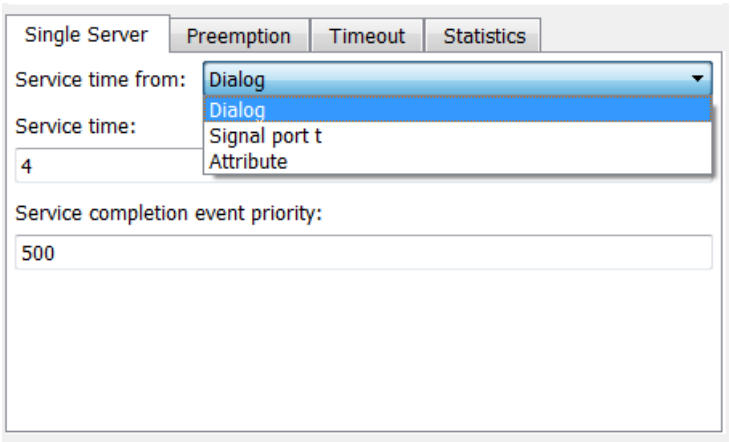
Tabulka 3.1: Popis distribucí bloku *Time-Based Entity generator*

Distribuce	**Popis**
Konstantní	Perioda generování entit se s časem nemění
Uniformní	Zadá se minimální a maximální rozmezí. V tomto rozmezí jsou entity generovány náhodně podle rovnoměrného rozdělení.
Exponenciální	Entity se generují náhodně podle exponenciálního rozdělení se zadanou průměrnou intenzitou.



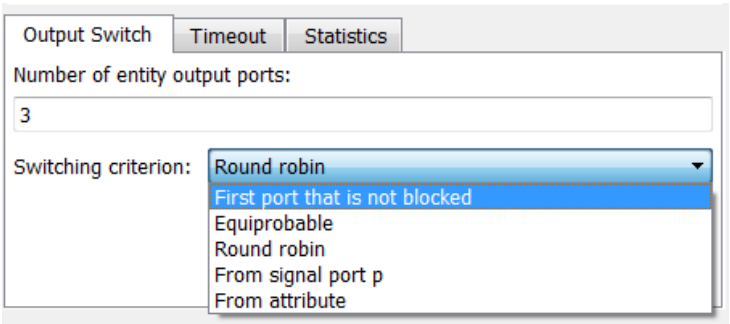
Obrázek 3.2: Nastavení bloku *Time-Based Entity generator*

Blok *Single Server* modeluje proces, který je na entitě prováděn po dobu procesního času. My tento blok používáme jako stroj výrobní linky, který dokáže pracovat pouze na jedné entitě zároveň. Pokud bychom měli například případ, že reálný stroj dokáže pracovat na čtyřech entitách zároveň, pak SimEvents nabízí použití bloku *N-Server*. Nastavení parametrů vidíme na obrázku 3.3. Hlavním parametrem serveru je Service Time, který představuje procesní čas. Tento čas lze pevně zadat pro každou entitu. Nebo lze tuto hodnotu měnit podle atributu entity anebo opět brát z portu.



Obrázek 3.3: Nastavení bloku *Single Server*

U zapojení serverů do paralelní sítě je potřeba použit rozbočovač *Output Switch*. Tabulku nastavení tohoto bloku ilustruje obrázek 3.4. Na tomto rozbočovači je samozřejmě nutné nastavit počet výstupních portů a také logiku přiřazování entit do paralelních větví. SimEvents nabízí možnost *Round robin*, která představuje naší logiku střídání. Nebo je nabídnuto rozřazování na základě atributů entit, vstupního signálu atd. Pro spojení paralelních větví do jedné můžeme použít blok *Path Combiner* nebo *Input Switch*, které se chovají podobně.

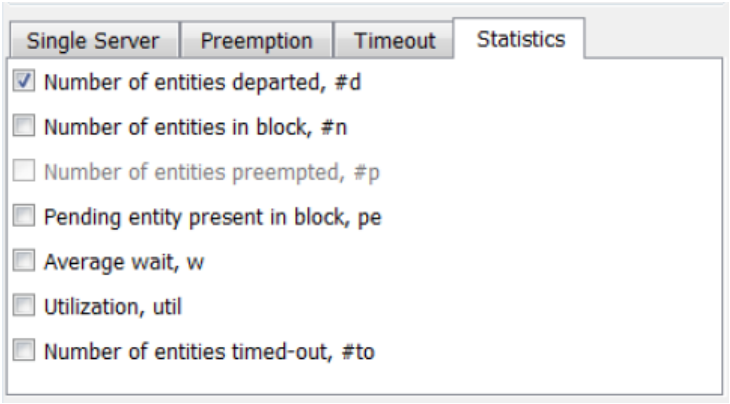


Obrázek 3.4: Nastavení bloku *OutputSwitch*

Blok *FIFO Queue*, který je na obrázku 3.1 slouží k informování o počtu čekajících entit před zpracováním. Tento blok je vhodné zapojit před servery nebo tam, kde by jsme mohli čekat tvoření front. Protože tento blok má v názvu FIFO, tak jistě tušíme, že entity se budou ve frontě chovat podle tohoto pravidla. V SimEvents je možné i použití *LIFO queue* bloku. Rozdíli mezi těmito pravidly jsme vysvětlili v kapitole 1. Tyto bloky potřebují jediný parametr a to je kapacita fronty. Po překročení této hodnoty se činnosti před frontou zastaví, a musí se čekat na uvolnění místa ve frontě. Například pro simulaci obchodu, kde servery tvoří pokladny a entity zákazníky, lze pro frontu nastavit dobu, po které zákazníkům dojde trpělivost a začnou odcházet.

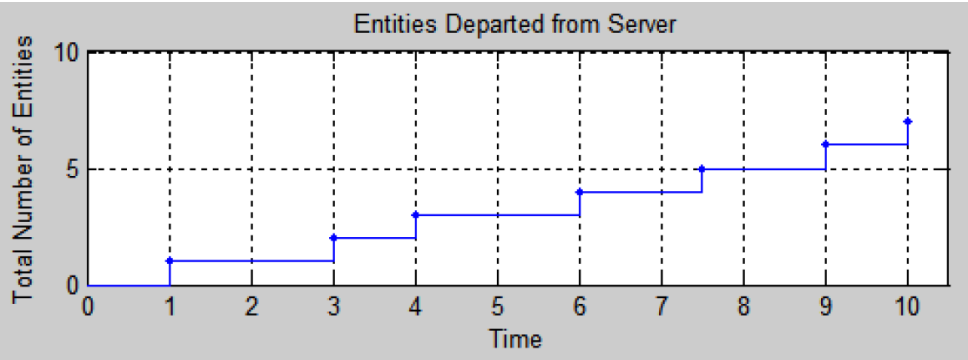
Pro zakončení série bloků používáme blok *Entity sink*, ve kterém entity končí a nikam dále nepokračují. U tohoto bloku je vhodné zjistit závislost počtu přijatých entit v čase.

Abychom mohli navrhnutý systém analyzovat, potřebujeme vizualizovat některá data z bloků. Většina bloků nabízí zpřístupnění portů, které posílají z bloku požadované statistiky ukázané na obrázku 3.5. Statistiky mohou být například počet entit v bloku v čase, časy odchodu entit z bloku, průměrná doba čekání atd. Porty lze napojit na blok *Signal Scope*, který na konci simulace zobrazí graf požadované statistiky. Počet odchozích entit v čase například ukazuje obrázek 3.6.



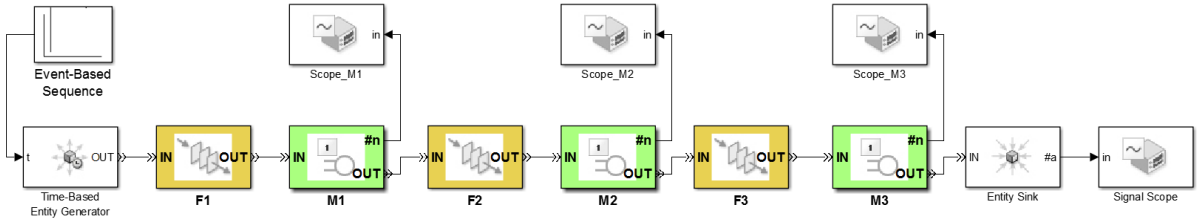
Obrázek 3.5: Statistika bloků

Program obsahuje širokou škálu různých bloků a nastavení. Tady jsme uvedli pouze základní, které budeme používat. Na obrázku 3.7 vidíme zapojení sériové výrobní linky z příkladu 2.1. Systém obsahuje bloky, které jsme v této kapitole popsali. Navíc lze zde vidět blok *Event-Based Sequence*, který je připojený na generátor entit. Tento blok má pouze za úkol, řídit počet vygenerovaných entit. V kapitole 2 jsme pokaždé pracovali s určitým počtem entit. Proto tedy používáme zmíněný blok, abychom do simulace vygenerovali například pouze 6 entit. Pro spuštění simulace pouze zbývá určit délku simulování. Protože jsme omezeni většinou na menší počet entit, odhadneme dobu simulování,

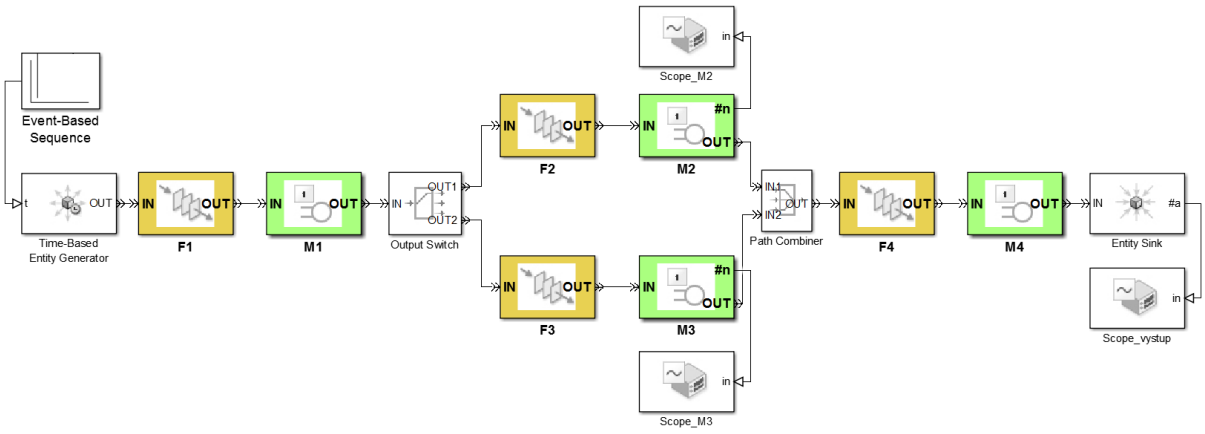


Obrázek 3.6: Ukázka grafu odchodu entit z bloku v závislosti na čase

aby nám stačila na projití všech vygenerovaných entit až na konec výrobní linky. Dále uvádíme na obrázku 3.8 zapojení paralelní výrobní linky z příkladu 2.2. Tady byl navíc přidán rozbočovač a blok pro spojení paralelních větví.



Obrázek 3.7: SimEvents - Sériová výrobní linka



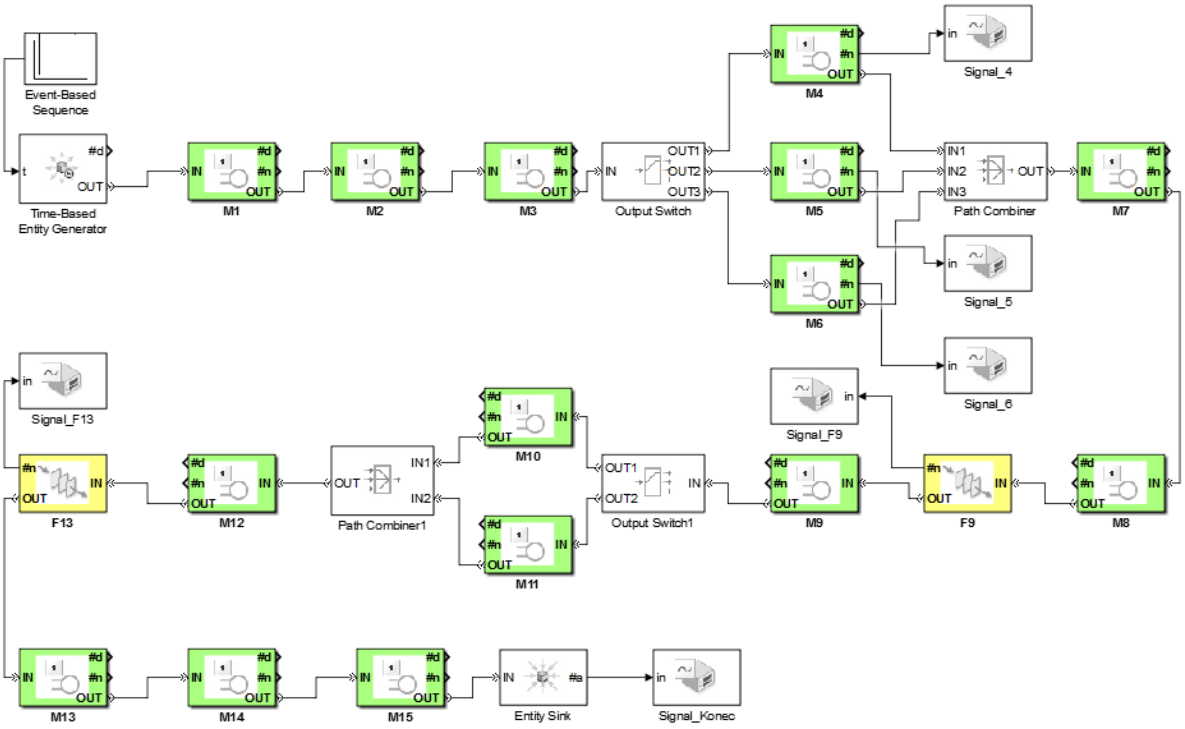
Obrázek 3.8: SimEvents - Paralelní výrobní linka

3.1 Obecná výrobní linka – simulace a porovnání

V této části vytvoříme simulace příkladů z kapitoly 2 a budeme výsledky porovnávat s Ganttovými diagramy ze stejné kapitoly. K vytvoření simulací využijeme zmíněný program SimEvents. Začneme rovnou příkladem z kapitoly 2.5. Jedná se tedy o příklad obecné výrobní linky s 15-ti stroji a uvažujeme fronty F9 a F13 zapojené podle obrázku 2.13.

Příklad 9

K vytvoření simulace budeme potřebovat 15 serverů, 2 fronty a několik rozbočovačů. Dále k některým důležitým serverům připojíme bloky, které nám graficky zobrazí počet entit ve stroji. Obrázek 3.9 zobrazuje navržený příklad v prostředí SimEvents. Stroje jsou značeny stejně jako v minulých kapitolách. Bloky, které zobrazí grafické výsledky byly připojeny ke strojům M4, M5 a M6. Další tento blok je připojen k výstupu výrobní linky a k frontám F9 a F13.

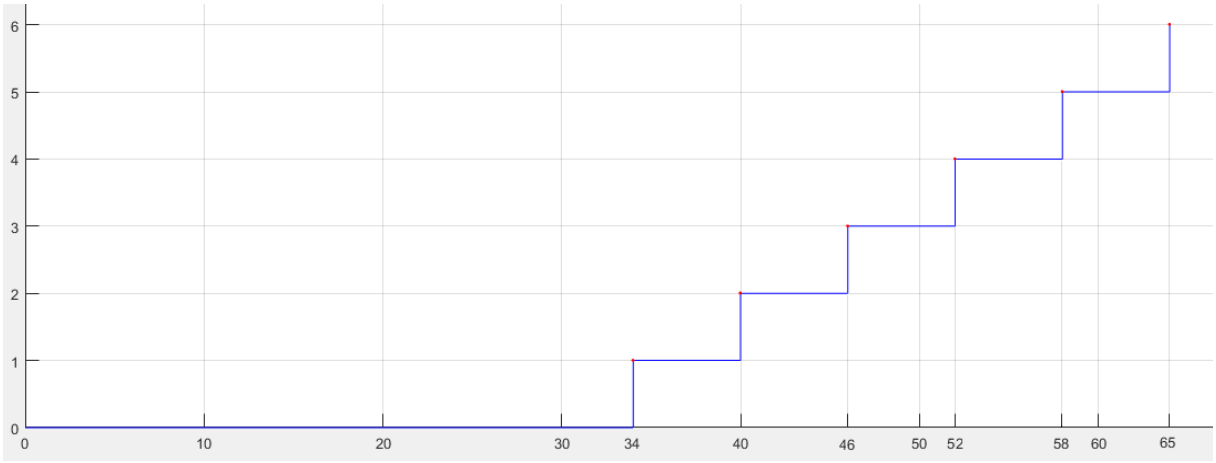


Obrázek 3.9: Příklad obecné výrobní linky v prostředí SimEvents

Generátor entit je nastaven tak, že posílá entity ihned, jak je stroj M1 volný. Pro náš případ generujeme pouze šest entit. Dále jsme nastavili procesní časy všem serverům podle příkladu z kapitoly 2.4. Důležité je zadat rozbočovačům do paralelních strojů jejích kritéria rozřazování. Zvolili jsme kritérium Round robin, které se chová podle logiky střídání.

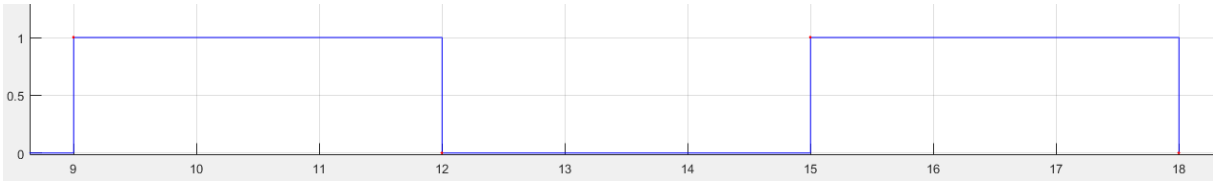
Pro správné chování a zobrazení počtů entit ve stroji jsme vložili blok fronty s chováním FIFO. Čas konce simulace jsme nastavili na 70 časových jednotek. Delší simulační čas není pro 6 entit potřeba. Po spuštění simulace jsme dostali požadované grafické výsledky o entitách ve strojích, frontách a na výstupu výrobní linky.

Na obrázku 3.10 vidíme závislost počtu entit na čase na výstupu výrobní linky. Můžeme si například všimnout, že v čase 34 první entita opustila výrobní linku. Dále můžeme vidět časy dalších entit. Z grafu lze poznamenat, že výrobní linkou proběhlo 6 entit za čas 65. Když se podíváme na Ganttův diagram 2.12 a přesněji na konečné časy procesů entit na M15, tak se tyto časy shodují s časy z obrázku 3.10. Tím jsme zjistili, že simulace z hlediska výstupních časů je správná.



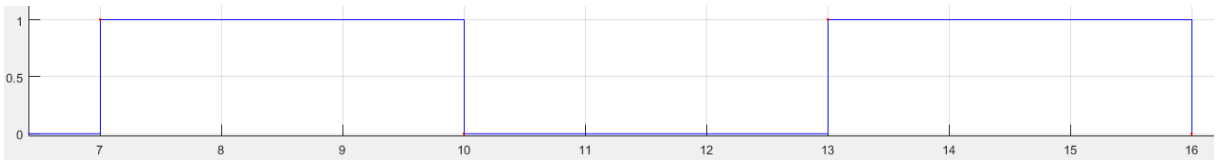
Obrázek 3.10: Počet entit v čase na výstupu výrobní linky

Dále se podíváme na výsledky z paralelních strojů M4, M5 a M6. Grafické výsledky ilustrují obrázky 3.11, 3.12 a 3.13. Například pokud se podíváme na obrázek 3.11, zjistíme, že ve stroji byla entita od času 5 do 12 a poté jiná entita od času 15 do 18. Podobné výsledky najdeme na ostatních obrázcích a pokud se podíváme na Ganttův diagram 2.12, můžeme říci, že se časy opět shodují. Co nelze z obrázků vidět je to, o kolikátou entitu se jedná. Tuhle informaci však pro tak málo entit nepotřebujeme ověřovat.

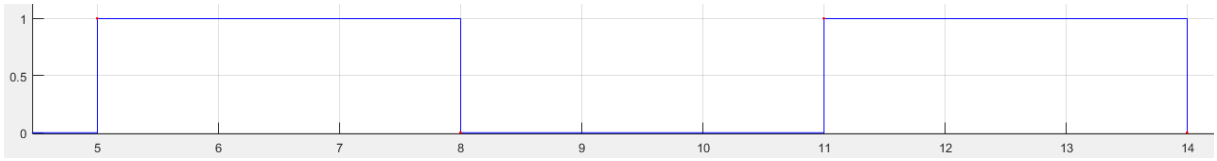


Obrázek 3.11: Počet entit v čase na stroji M4

Grafické výstupy námi zvolených strojů jsme si vybrali z toho důvodu, že jsou pro nás nejzajímavější a nejvíce nám potvrdí správnost vytvořené simulace. Pro ostatní stroje



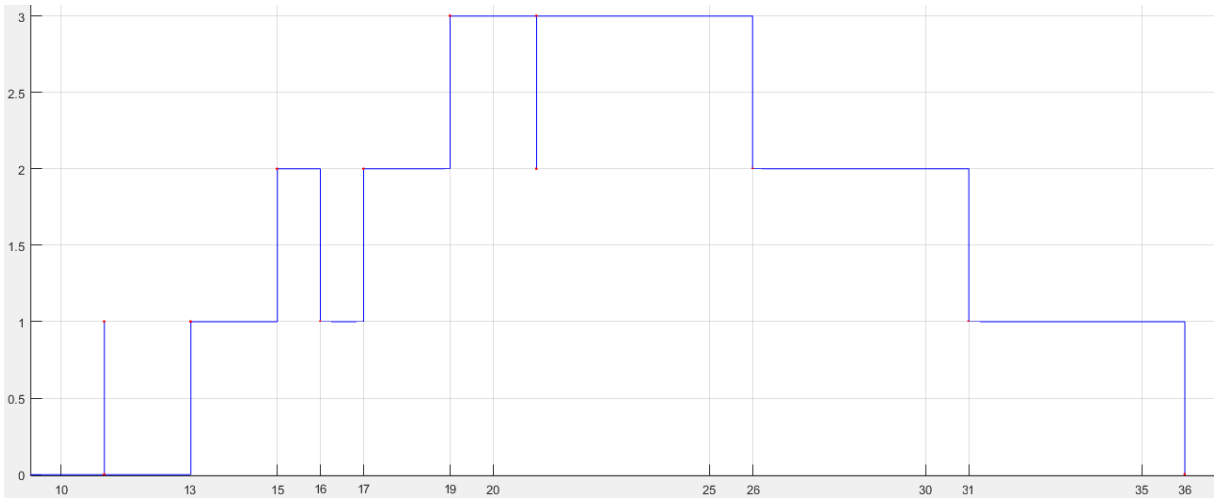
Obrázek 3.12: Počet entit v čase na stroji M5



Obrázek 3.13: Počet entit v čase na stroji M6

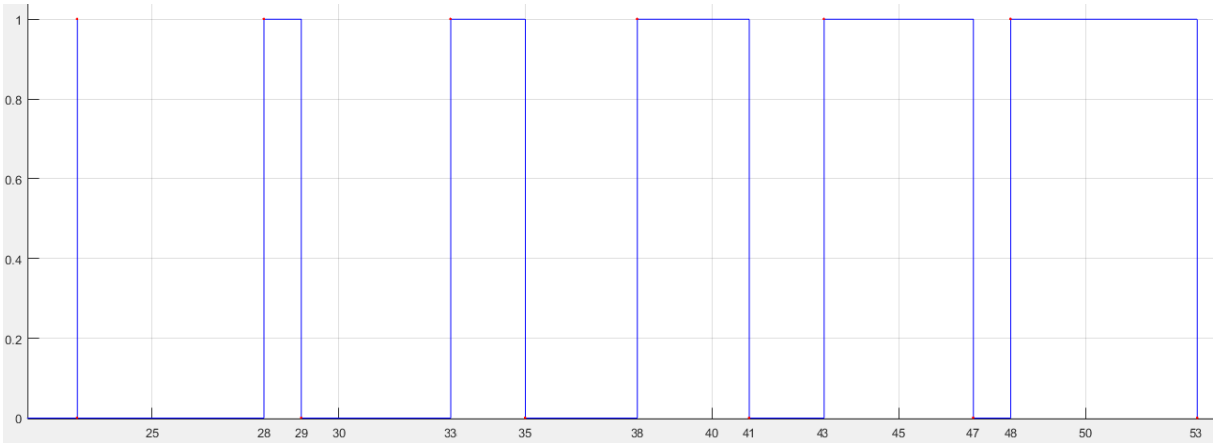
bychom si mohli také vyžádat grafické výstupy pro všeobecnou kontrolu. To bychom však nechali pro podrobnější analýzy a složitější příklady. Grafickým výstupem nemusí být pouze počet entit ve stroji, ale také čas odchodů entit ze strojů. My jsme však zvolili počet entit pro výstižnější prezentaci výsledků.

Na řadě máme kontrolu front F9 a F13. Grafické výstupy z front vidíme na obrázcích 3.14 a 3.15. Tyto výstupy opět ukazují počet entit ve frontě v daných časech. Porovnáním z obrázkem 2.14 opět vidíme shodu. Můžeme tedy říci za předpokladu, že se simulace chová podle skutečnosti, že matematický model je správně sestaven.



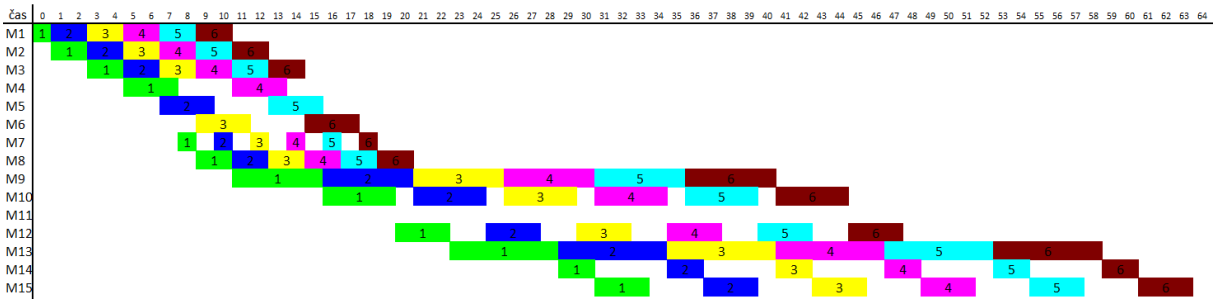
Obrázek 3.14: Počet entit v čase ve frontě F9

Přece jenom uvedené výstupní grafy ze simulace se špatně prezentují. Nejvýstižnější výstup je Ganttův diagram, který v kapitole 2 byl často používán a dával úplnou informaci o procesech na strojích. Pokud do simulace přidáme tzv. *To Workspace blok*, můžeme tyto bloky napojit na výstupní porty všech strojů. Těmito porty necháme přenášet infor-



Obrázek 3.15: Počet entit v čase ve frontě F13

maci o počtu entit v čase. Poté přes *To workspace blok* přeneseme data do matlabovského prostředí, kde s nimi můžeme pracovat. V Matlabu data zpracujeme do požadované matice, která bude vhodná pro export to tabulky v programu MS Excel, který používáme pro vytváření Ganttová diagramu pomocí naprogramovaného makra. Obrázek 3.16 ilustruje Ganttův diagram simulované výrobní linky. Výsledkem je, že výrobní linkou projde 6 entit za 64 časových dílků. Tento diagram je shodný s diagramem 2.14. Tyto diagramy se jenom liší v uspořádaní procesů na paralelních strojích. Simulaci v programu SimEvents najdeme v příloze pod názvem *vyr_linka.slx*.

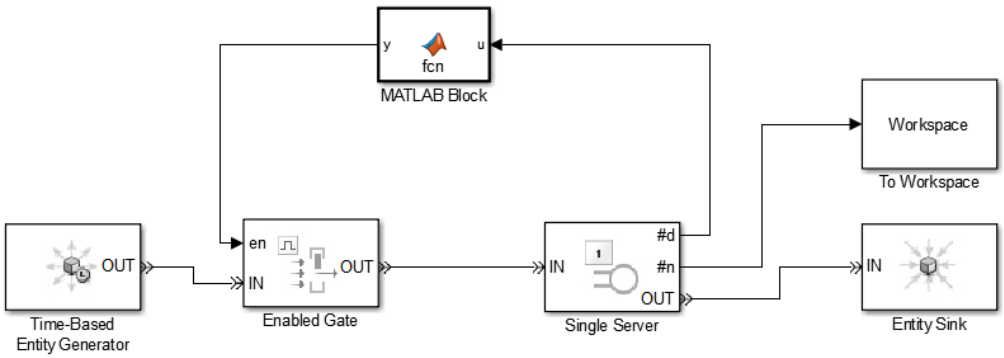


Obrázek 3.16: Ganttův diagram výrobní linky-simulační přístup

3.2 Heuristický přístup

V této kapitole budeme řešit plánování optimálních údržeb stejného zadání jako v kapitole 2.6. Tentokrát se, ale zaměříme na simulační přístup jako v předchozí kapitole 3.1. Program SimEvents sám nenabízí řešení této úlohy. Je potřeba zakomponování tzv. *Matlab bloků*, které zahrnou do simulace vlastní matlabovské kódy. Do těchto bloků, lze

posílat statistiky z jiných bloků. Jak jsme se už zmínili, statistikami mohou být například údaje o tom, kolik je právě v bloku entit atd. Tyto údaje potom uložíme do proměnných a pomocí matlabovského programování můžeme s těmito údaji pracovat. Poté opět lze upravené nebo nové údaje ve formě proměnných posílat do bloků, které zpracovávají tyto informace ze vstupních portů. Například můžeme takto upravovat procesní časy na serverech nebo pozastavovat server. To se provádí pomocí bloku *Enabled Gate*, který pozastavuje průchod entit na základě binárních hodnot ze vstupního portu. Na obrázku 3.17 vidíme ukázku zapojení zmiňovaných bloků.

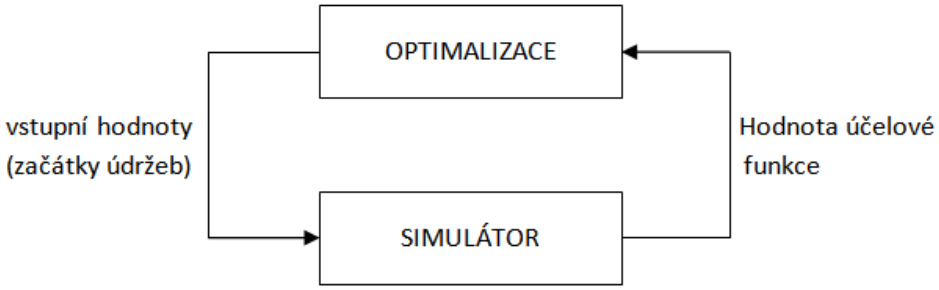


Obrázek 3.17: Ukázka zapojení bloků

Prozatím jsme vysvětlili, jak simulaci v SimEventu rozšířit o další funkce. Teď je potřeba, abychom se zabývali algoritmy, které řeší optimalizaci údržeb na výrobní lince. Tyto algoritmy poté implementujeme do matlabovského kódu, který bude použit v *Maltab bloku*. Algoritmus, který pro naší úlohu použijeme, patří do skupiny heuristických algoritmů. Na obrázku 3.18 můžeme vidět schéma principu heuristického přístupu. Simulátor bereme jako černou skříňku, do které vkládáme data. Data, která ze skříňky vycházejí optimalizujeme a poté opět vkládáme do simulátoru. Proces se poté opakuje, dokud nedostaneme hledané řešení.

Mezi základní algoritmy lze řadit metodu pokus omyl, která slouží jako příklad. Spíše se využívají pokročilejší metody. Jestliže chceme hledat řešení podle strategie, tak říkáme, že používáme heuristiku. Je však důležité říci, že nejsou zaručena optimální řešení. Často je však užitečné zjistit alespoň řešení lepší než dosud nejlepší známé.

Zdroj [34] uvádí popis nejznámějších heristických algoritmů. Příkladem jsou genetický, netopýří algoritmus nebo optimalizace hejnem světlušek. Zdroj [26] uvádí, že na úlohy plánování má optimalizace hejnem světlušek (firefly algorithm) lepší vlastnosti než jiné algoritmy, proto tento algoritmus použijeme pro naši úlohu. Skutečné světlušky se cho-



Obrázek 3.18: Schéma heuristického přístupu

vají tak, že svítí určitou frekvencí a jasností. Jasnost se snižuje se zvyšující vzdáleností od zdroje. Jasnost závisí také na absorpci světla okolí. Světlušky se chovají tak, že frekvencí a intenzitou svého světla přitahují ostatní světlušky opačného pohlaví. Vlastnost jak moc světluška přitahuje jinou budeme nazývat atraktivnost. Tento druh chování je základ optimalizace hejnem světlušek, který vznikl v roce 2007 a je popsán ve zdroji [34]. Tento algoritmus používá tato pravidla:

- Každá světluška může být světélkem atraktivní pro jakoukoliv jinou světlušku bez závislosti na pohlaví.
- Atraktivnost je závislá na intenzitě světla. Pro každé dvě světlušky bude platit, že ta s menší intenzitou světla se bude pohybovat k té s větší intenzitou světla. Atraktivnost a intenzita světla se snižuje s rostoucí vzdáleností. Pokud ani jedna světluška nesvítí jasněji, tak potom se pohybují náhodně.
- Intenzita světla je určena účelovou funkcí.

Algoritmus 1 Pseudo kód algoritmu hejnem světlušek

1: *Účelová funkce* $f(\mathbf{x})$, $\mathbf{x} = (x_1, \ldots, x_d)$, *kde* d *je dimenze*
2: *Generuj počáteční populaci světlušek* $\mathbf{x}_i$, *kde* $i = 1, 2, \ldots,$ n *a* n *je počet světlušek*
3: *Intenzita světla* I_i *v* $\mathbf{x}_i$ *je určena funkcí* $f(\mathbf{x}_i)$
4: *Urči koeficient světelné absorpce* γ
5: **while** $t <$ *maximální počet iterací* **do**
6: **for** $i = 1 : n$ **do**
7: **for** $i = 1 : n$ **do**
8: **if** $I_i < I_j$ **then**
9: *Světluška i se bude pohybovat k světlušce j*
 end if
10: *Změn atraktivitu se vzdáleností r přes* $exp[-\gamma r]$
11: *Přepočitej nové řešení a uprav intenzitu světla* I_i
 end for j
 end for i
12: *Seřaď světlušky a najdi současné globální extrém* **g**
 end while
13: *Vizualizuj výsledky*

Pseudo kód je ukázán v algoritmu 1. Vektor $\mathbf{x}_i$ představuje polohový vektor světlušky i. Hodnotou r_{ij} označíme vzdálenost světlušky i a j a její výpočet provedeme euklidovskou normou (3.1).

$$r_{ij} = ||\mathbf{x}_i - \mathbf{x}_j|| = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2} \tag{3.1}$$

Algoritmus ve svém výpočtu používá několik parametrů, které ovlivňují výpočet, a které si zadefinujeme. Absorpční koeficient γ vyjadřuje vlastnost okolí absorbovat světlo. Tento parametr použijeme v rovnici (3.2), kde intenzita světla je vyjádřena funkcí $e^{-\gamma r^m}$, která závisí na vzdálenosti r. Hodnota β_0 je atraktivnost ve vzdálenosti $r = 0$ a β je atraktivnost ve vzdálenosti r. Parametrem m zrychlujeme úbytek intenzity světla v závislosti na vzdálenosti.

$$\beta = \beta_0 e^{-\gamma r^m}, \ m \geq 1 \tag{3.2}$$

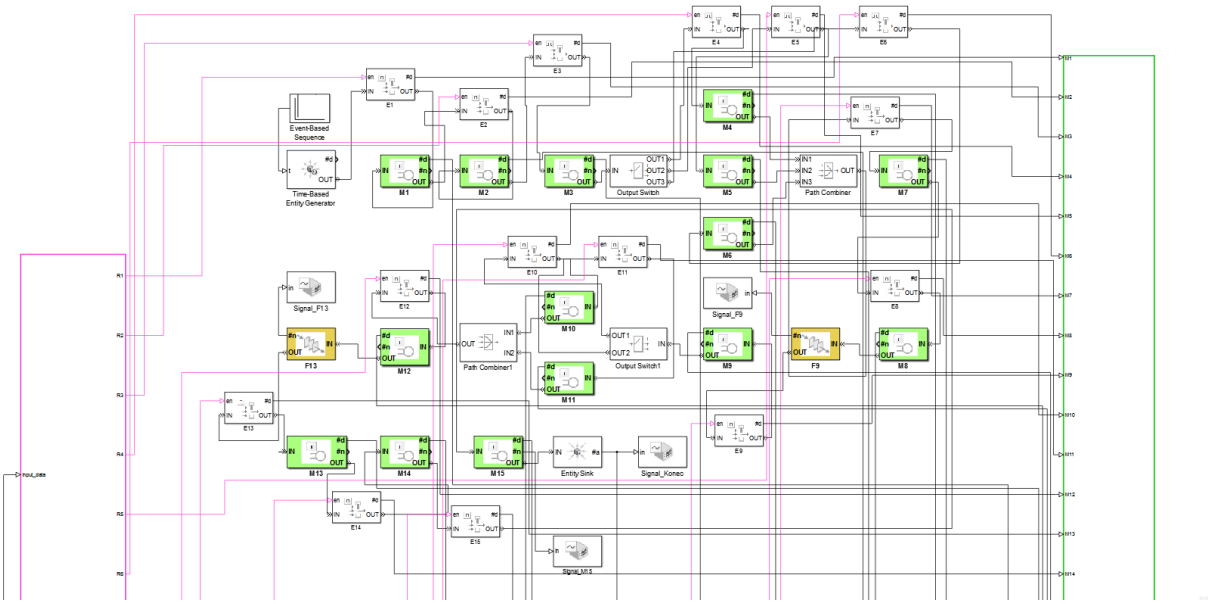
Pohyb světlušky i k více atraktivní světlušce j je určen vztahem (3.3). Ten říká, že nový polohový vektor světlušky i získáme ze starého polohového vektoru, atraktivity a náhodného vektoru ϵ_i. Vektor ϵ_i je vektor náhodných čísel podle Gaussovského nebo

rovnoměrného rozdělení. Parametr $\alpha \in [0, 1]$ je zvolen podle toho, jak moc chceme, aby poloha byla ovlivněna náhodou. Pokud volíme $\beta_0 = 0$, pak dostáváme náhodnou procházku. Náhodná procházka je proces, kdy novou hodnotu získáme sečtením předchozí hodnoty a náhodné hodnoty. O náhodné procházce se můžeme dočíst ve zdroji [34].

$$\mathbf{x}_i = \mathbf{x}_i + \beta_0 e^{-\gamma r_{ij}^m}(\mathbf{x}_j - \mathbf{x}_i) + \alpha\epsilon_i \tag{3.3}$$

Příklad 10

V tomto příkladu použijeme stejně navrženou simulaci výrobní linky jako v příkladu 9. Do tohoto navržení v programu SimEvents potřebujeme přidat nové bloky. Zejména *Enabled Gate*, který na základě binárního signálu zavírá nebo otevírá průchod entit. Dále přidáme blok *To Workspace*, který nám zpracuje signály do matlabovského prostředí. Signály, se kterými budeme pracovat jsou časy odchodů entit ze strojů. Na obrázku 3.19 vidíme větší část zapojení bloků.



Obrázek 3.19: SimEvents - zapojení pro plánování údržeb

Nyní budeme chtít pomocí optimalizace hejnem světlušek spočítat optimální údržby se stejným zadáním jako v kapitole 2.6. Vytvoříme si matlabovský m-file, ve kterém implementujeme optimalizaci. Světlušky budou představovat vektory, jejíchž složky znamenají začátky časů údržeb na každém stroji. V tomto případě vektory budou mít 15 složek. Vstupními parametry pro výpočet jsou: počet světlušek, počet iterací, m, α a β. Význam těchto parametrů jsme již uvedli a volbu těchto parametrů uvádí tabulka 3.2.

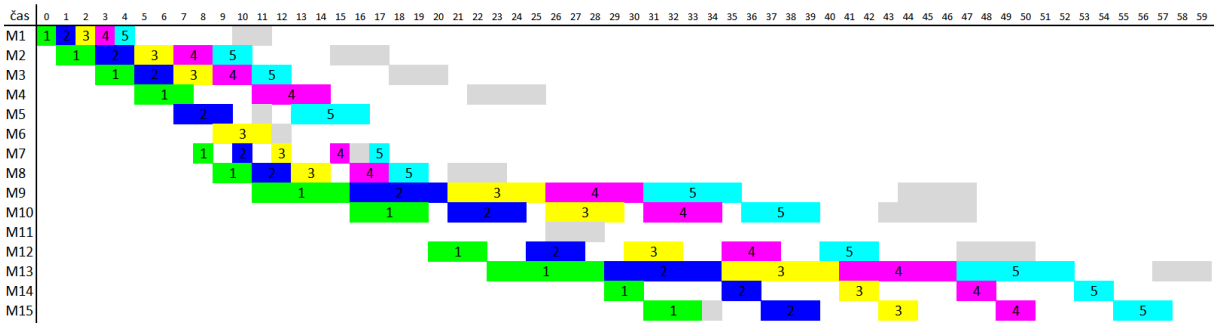
Parametr γ jsme volili pro každý stroj jiný v závislosti na délce údržby.

Tabulka 3.2: Tabulka parametrů pro algoritmus hejnem světlušek

Parametr	**Hodnota**
počet světlušek	20
počet iterací	100
m	2
α	1
β	0,1

Nevolíme účelovou funkci (2.3) jako doposud, ale tentokrát budeme minimalizovat pouze odchozí časy entit na posledních strojů. Důvod je takový, že minimalizovat všechny časy by bylo výpočetně náročné. Nová účelová funkce (3.4) má tu vlastnost, že se snaží minimalizovat dokončení jednotlivých entit na výrobní lince, ale nezaručuje, že entity budou na strojích M1 až M14 opracovány co nejdříve. Tento přístup pro princip hledání optimálních údržeb bude dostačující. Funkcí atraktivnosti světlušek tedy bude nově zvolená účelová funkce. Algoritmus pracuje s daty ze simulace, kterou po každé iteraci znovu spouští, aby získal nová data a porovnal účelové funkce. Po stech iteracích v tomto případů dochází k tomu, že světlušky mají účelové funkce stejné nebo jejich rozdíl je malý. Dostáváme, ale 20 různých řešení a my použijeme to s nejmenšími hodnotami. Potom data převedeme do požadované matice, kterou zpracujeme v programu MS Excel a vykreslíme Ganttův diagram 3.20.

$$\min \sum_{i=1}^{n} x_{i,6} \tag{3.4}$$



Obrázek 3.20: Ganttův diagram pro hledání optimálních údržeb pomocí heuristiky

Na diagramu vidíme, že výrobní linkou prošlo 5 entit za dobu 58 časových dílků. To je stejný výsledek jako na obrázku 2.18, i přesto že se uspořádání entit na paralelních strojích

malinko liší a údržby se překrývají. Hlavní věc, které si všimneme je, že časy údržeb jsou rozdílné. To je způsobeno tím, že v přístupu matematického modelování, jsme minimalizovali i časy údržeb. V simulačním přístupu tomu tak není. Důležitou příčinou však je, že v optimalizaci hejnem světlušek pracujeme s náhodností. Tedy hledáme výsledek náhodně a není zaručeno, že výsledek je optimální a také že je jediný. Řešení je tedy více a máme jich k dispozici tolik, kolik jsme na začátku výpočtu zvolili počet světlušek. My nemusíme na konci procesu zvolit světlušku s nejmenšími hodnotami, ale i jinou, která nám vyhovuje podle požadavků. Výsledek závisí také na změně parametrů. Je tedy potřeba s parametry více pracovat a je nepravděpodobné, že na první pokus dostaneme hned výsledky. U přístupu matematického programování jsme zkoušeli vyřešit podmínku, aby se údržby nepřekrývaly. To by šlo řešit zavedením penalizační funkce, která bude v případě překrytí údržby zvyšovat hodnotu účelové funkce, a tím i zhorší kvalitu řešení. O tomto principu se můžeme dočíst ve zdroji [33]. Účelová funkce (3.4) by byla upravena na účelovou funkci (3.5), která obsahuje parametr Q a indikátorovou proměnnou $z_{j,j'}$, kde $j = 1, \ldots, \ m$ a $j' = 1, \ldots, \ m$.

$$\min \left(\sum_{i=1}^{n} x_{i,6} + Q \sum_{j=1}^{m} \sum_{j'=1}^{m} z_{j,j'} \right) \tag{3.5}$$

Parametr Q volíme jako velké vhodné číslo a $z_{j,j'}$ se řídí předpisem (3.6).

$$z_{j,j'} = \begin{cases} 1 & \text{pokud údržba na stroji } j \text{ se překrývá s údržbou na stroji } j' \\ 0 & \text{jinak} \end{cases} \tag{3.6}$$

Kapitola 4

Analýza výpočtů a porovnání

4.1 Analýza výpočtů matematického modelu

V této kapitole se budeme zabývat výpočtovou náročností příkladů z předchozích kapitol. Dále bychom chtěli nalézt omezení počtu entit a strojů.

Tabulka 4.1 obsahuje časy trvání výpočtu pro různé počty entit. Počet strojů jsme zvolili 15 a analýza je prováděna na příkladech z předchozích kapitol. Výpočty byly prováděny na počítači s procesorem Intel Core i5-2450M CPU 2.50GHz s operační pamětí 8GB. Znak * znamená, že výpočet nebyl realizován z důvodu velké délky výpočtového času.

Tabulka 4.1: Tabulka výpočtových časů matematických modelů

Model/Počet entit	**6**	**10**	**11**	**15**	**1000**
Sériová výrobní linka	0,02s	0,02s	0,02s	0,02s	3,32s
Paralelní výrobní linka - LP	0,48s	0,52s	0,52s	0,6s	8581s
Obecná výrobní linka	0,09s	54s	2399s	více než 24h	*
Obecná výrobní linka-fronty	1,95s	2028s	3365s	více než 24h	*
Obecná výrobní linka-údržby	3,96s	53,62s	2368s	více než 24h	*

Z analýzy jsme zjistili, že pracovat s modelem sériové linky není výpočtově náročné. Tento model dokáže pracovat až s tisící entitami stále relativně rychle. Tento počet entit můžeme pokládat za dostačující. U modelu čistě paralelního dostáváme pro méně entit o něco vyšší výsledky, avšak model s tisíci entitami počítáme podstatně déle než u modelu sériové linky. Analýzu paralelního modelu jsme prováděli pouze pro lineární verzi. Nelineární variantu jsme nezahrnuli, protože je u ní velice obtížné měnit počet entit. U obecného modelu je výpočtová náročnost vysoká, a je to způsobeno paralelními částmi a počtem strojů. Tyto části potřebují zavést binární proměnné, a potom se problém stává neceločíselný. Do deseti entit model dokáže být relativně spočítán rychle. Avšak zvýšením entit o jednu už čas výpočtu naroste a výpočet trvá několik hodin. Pokud je tedy model použitelný pouze do deseti entit, není potom moc použitelný pro větší práci. V málokteré

výrobní lince se vyrábí například 10 kusů za den. Model by se dal jedině použit při tom, kdybychom entity brali jako zakázky a dívali bychom se spíš na výrobu globálně. Tedy například by jedna entita znamenala vyrábět vláčky, druhá entita vyrábět autíčka atd. Takže pro menší plánování by se dal model univerzálně použít. Je však možné, že model lze různými triky upravit nebo ho spočítat na výkonnějším počítači. Také jinou volbou účelové funkce bychom mohli dosáhnout jiných výpočtových časů. Jak ale uvádí [31], okruh těchto úloh řadíme do NP těžkých úloh.

4.2 Analýza výpočtů simulačního přístupu a porovnání

V této kapitole uvedeme výpočtovou náročnost simulací, které jsme provedli v kapitole 3.1. Tyto hodnoty porovnáme s výpočtovou náročností matematických modelů výrobních linek z kapitoly 2. Hodnoty časů simulačního přístupu uvádí tabulka 4.2. Analýza byla provedena za stejných podmínek, které byly uvedeny v kapitole 4.1.

Tabulka 4.2: Tabulka porovnání výpočtových časů

Model/Počet entit	**6**	**15**	**1000**
Simulace - Obecná výrobní linka-fronty	4s	5s	8s
Simulace - Obecná výrobní linka-údržby	141s	165s	19317s

Jak jsme zřejmě tušili, výpočtová náročnost simulací je mnohem menší. Můžeme to však říct pouze pro příklady, které jsme počítali. Simulováním složitějších systémů s hodně rozhodovacími kroky v podobě heuristiky a zapojením matlabovských kódů se výpočtová náročnost simulace zvyšuje. Avšak stále je únosnější, než pro počítání pomocí matematického programování. Používání programu SimEvents je však pohodlné a k jeho používání není nutná velká znalost matematiky. Každá změna, kterou je potřeba provést, například přidání entit, strojů atd., je velice snadná a rychlá. Nevýhodou je, že SimEvents nenabízí řešení plánování údržeb a optimalizaci, a je potřeba doprogramování. Tím jsou možnosti omezeny na umění programátora. Přidávání bloků a jejich zapojení se nemusí řešit pouze ručně, ale lze napsat kód, který celý systém vygeneruje s požadovaným nastavením. To je velice užitečné, ale bohužel neexistuje k tomuto postupu obsáhlý návod od výrobců SimEvents. Takže není jisté co vše lze kódem vytvořit, protože pro to neexistuje návod. Tabulka 4.3 uvádí ceny pro komerční zakoupení produktu, které jsou potřeba pro práci s modely matematického programování nebo simulace těchto modelů. Řešič BARON je potřeba pro výpočet úloh MINLP a NLP zatímco řešič CPLEX dokáže počítat úlohy

LP a MIP. Pro simulaci v programu SimEvents je zapotřebí koupě produktů MATLAB, Simulink a SimEvents.

Tabulka 4.3: Ceny použitých produktů

Produkt	**Cena [Euro]**
GAMS/BARON	2 900
GAMS/CPLEX	8 750
MATLAB	2 000
SIMULINK	3 000
SIMEVENTS	2 850

Závěr

Tato práce vznikla na základě spolupráce s firmou BOSCH DIESEL s.r.o. - Jihlava, která požadovala vytvoření simulace výrobní linky. Firma v prvním kroku nepotřebovala optimalizovat linku, ale pouze mít simulační nástroj, se kterým by pracovala při plánování odstávek. Simulační model byl realizován v programu SimEvents.

V diplomové práci byly shrnuty potřebné poznatky operačního výzkumu. V části matematického modelování bylo sestaveno celkem 7 modelů různých typů výrobních linek. Tyto modely byly různě klasifikovány a řešeny různými metodami. Ke každému modelu byl uveden příklad, který byl implementován v programu GAMS. V další části práce byl rozpracován simulační přístup modelování výrobních linek za použití programu SimEvents. V této kapitole byl simulován příklad obecné výrobní linky s frontami. Na tomto příkladě jsme se snažili řešit plánování údržeb pomocí heuristických algoritmů. Tato metoda se ukázala jako použitelná pro tento problém, přestože není zaručeno optimálních výsledků. Proto je důležité s touto metodou individuálně pracovat pro různé případy a hlouběji analyzovat zadaný problém konkrétní výrobní linky. Dalšími výsledky této práce jsou výpočtové časy různých modelů a přístupů, které byly porovnány a zhodnoceny. Zjistily se výhody a nevýhody těchto dvou přístupů. Simulační přístup je mnohem snadnější a rychlejší než přístup matematického modelování, avšak je omezen na možnostech prostředí SimEvents. Velkým problémem u matematického přístupu je, že s rostoucím počtem entit roste velice časová náročnost výpočtu. Všechny výsledky byly prezentovaný Ganttovými diagramy v MS EXCEL, které byly generovány na základě výstupních dat z programu GAMS nebo SimEvents. Diagramy jsou vytvořeny pomocí makra a jazyka VBA. Je proto možné univerzálně používat soubor *vizualizace.xlsm*, který najdeme v přílohách. Z důvodu velkého objemu dat zdrojových kódů programu GAMS jsou tyto kódy zahrnuty v přílohách této práce v souborech typu *GMS*.

V celé práci jsme veličiny a parametry v matematickém programování, ale i v simulačním přístupu, uvažovali deterministické. V některých případech by mohl být tento přístup správný a náhodné vlivy bychom mohli zanedbat. Parametry t_j, které představují doby trvání procesů na jednotlivých strojích, ale nemusí být vždy stejné a mohou být náhodné. Snadnější však bude pracovat s průměrnými hodnoty z časů, které jsme například vícekrát naměřili. Další náhodné jevy bychom mohli zavést v řazení entit do paralelních

strojů nebo v generování entit na začátku výrobní linky. Poznatky o stochastických jevech ve výrobních linkám nalezneme ve zdroji [24]. Dále bychom mohli modely rozšířit o dopravníky mezi stroji, které jsme v tomto textu neuvažovali. Dopravník by se zavedl jako další stroj, jehož procesní čas by znamenal dobu entity strávenou na dopravníku. Doprava do paralelních strojů může být řešena portálem, který je naprogramován určitou logikou, aby entity rozřazoval. Cestování portálu bychom museli zahrnout a je pravděpodobné, že doba dopravy pro každou entitu by byla různá.

Velké výpočtové nároky matematického programování by se daly snížit paralelizací výpočtu. To znamená, že bychom výpočet rozdělili na více částí, které by byly řešeny více procesory v jednom čase. Metody, jak toto rozdělení provést najdeme v [23].

Práce by mohla sloužit k základnímu pochopení modelování výrobních linek. Nemusí se nutně používat pouze jednoho přístupu, ale je vhodné přístupy kombinovat. Používání simulace a matematického programování je perspektivní možností pro řešení problematiky výrobních linek.

Literatura

[1] ARROYO, José Elias C. a Joseph Y.-T. LEUNG. Scheduling unrelated parallel batch processing machines with non-identical job sizes and unequal ready times. Computers & Operations Research [online]. 2017, 78, 117-128 [cit. 2017-05-16]. DOI: 10.1016/j.cor.2016.08.015. ISSN 03050548. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S030505481630209X

[2] BAHR, H.A., R.F. DEMARA, John OYEKAN a Ashutosh TIWARI. Smart priority queue algorithms for self-optimizing event storage: New Opportunities and Future Trends. Simulation Modelling Practice and Theory. IEEE, 2004, 12(1), 15-40. DOI: 10.1016/j.simpat.2004.01.002. ISBN 978-1-4799-7486-3. ISSN 1569190x. Dostupné také z: http://linkinghub.elsevier.com/retrieve/pii/S1569190X04000036

[3] BAPAT, R. B. Graphs and matrices. New Delhi: Hindustan Book Agency, 2010. Universitext. ISBN 978-1-84882-980-0.

[4] BAZARAA, M. S., Hanif D. SHERALI a C. M. SHETTY. Nonlinear programming: theory and algorithms. 3rd ed. Hoboken: John Wiley, 2006. ISBN 978-0-471-48600-8.

[5] CLAASSEN, G.D.H., J.C. GERDESSEN, E.M.T. HENDRIX a J.G.A.J. VAN DER VORST. On production planning and scheduling in food processing industry: Modelling non-triangular setups andproduct decay. Computers & Operations Research [online]. 2016, 76, 147-154 [cit. 2017-05-16]. DOI: 10.1016/j.cor.2016.06.017. ISSN 03050548. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0305054816301496

[6] CLARK, Wallace. Ganttovy diagramy: pomůcka k řízení podniků a práce. Praha: Orbis, 1931. Organisace a řízení. Dostupné také z: http://www.digitalniknihovna.cz/mzk/uuid/uuid:609e94b0-7d09-11e3-9be6-005056827e52

[7] GOSAVI, Abhijit. Simulation-based optimization: parametric optimization techniques and reinforcement learning. Boston: Kluwer Academic Publishers, c2003. ISBN 1402074549.

[8] GRAVES, S. C., A. H. G. RINNOOY KAN a Paul Herbert. ZIPKIN. Logistics of production and inventory. New York: North-Holland, 1993. ISBN 978-0444874726.

[9] GRAY, Michael A. Discrete Event Simulation: A Review of SimEvents. Computing in Science. 2007, 9(6), 62-66. DOI: 10.1109/MCSE.2007.112. ISSN 1521-9615. Dostupné také z: http://ieeexplore.ieee.org/document/4362731/

[10] GROSS, Donald. Fundamentals of queueing theory. 4th ed. Hoboken, N.J.: Wiley, c2008. ISBN 978-0-471-79127-0.

[11] GROSS, Jonathan L. a Jay. YELLEN. Graph theory and its applications. Boca Raton, Fla.: CRC Press, c1999. ISBN 0-8493-3982-0.

[12] HAX, Arnoldo C. a Dan. CANDEA. Production and inventory management. Englewood Cliffs, N.J.: Prentice-Hall, c1984. ISBN 0137248806.

[13] HERTERICH, Matthias M., Falk UEBERNICKEL, Walter BRENNER a Ashutosh TIWARI. The Impact of Cyber-physical Systems on Industrial Services in Manufacturing: New Opportunities and Future Trends. Procedia CIRP. IEEE, 2015, 30(1), 323-328. DOI: 10.1016/j.procir.2015.02.110. ISBN 978-1-4799-7486-3. ISSN 22128271. Dostupné také z: http://linkinghub.elsevier.com/retrieve/pii/S2212827115001924

[14] HLINĚNÝ, Petr. Teorie grafů. 2008, 125 s. Dostupné také z: https://www.fi.muni.cz/ hlineny/Vyuka/GT/Grafy-text07.pdf

[15] [online]. In: . [cit. 2017-05-23]. Dostupné z: https://s-media-cache-ak0.pinimg.com/originals/c3/2b/39/c32b39ddef67c90f0c772b72f886f4a4.jpg

[16] JAIN, Sanjay a Guodong SHAO. Virtual factory revisited for manufacturing data analytics. Proceedings of the Winter Simulation Conference 2014. IEEE, 2014, , 887-898. DOI: 10.1109/WSC.2014.7019949. ISBN 978-1-4799-7486-3. Dostupné také z: http://ieeexplore.ieee.org/document/7019949/

[17] KAPLANOĞLU, Vahit. An object-oriented approach for multi-objective flexible job-shop scheduling problem. Expert Systems with Applications [online]. 2016, 45, 71-84 [cit. 2017-05-16]. DOI: 10.1016/j.eswa.2015.09.050. ISSN 09574174. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S095741741500679X

[18] KLAPKA, Jindřich. Metody operačního výzkumu. Vyd. 2. Brno: VUTIUM, 2001. ISBN 8021418397.

[19] KLEINROCK, Leonard. Queueing systems. New York: John Wiley, c1976. ISBN 0-471-49111-x.

[20] KUCK, Mirko, Jens EHM, Torsten HILDEBRANDT, Michael FREITAG a Enzo M. FRAZZON. Potential of data-driven simulation-based optimization for adaptive scheduling and control of dynamic manufacturing systems: New Opportunities and Future Trends. 2016 Winter Simulation Conference (WSC). IEEE, 2016, 38(1), 2820-2831.

DOI: 10.1109/WSC.2016.7822318. ISBN 978-1-5090-4486-3. ISSN 22128271. Dostupné také z: http://ieeexplore.ieee.org/document/7822318/

[21] LEE, Jay, Hossein Davari ARDAKANI, Shanhu YANG a Behrad BAGHERI. Industrial Big Data Analytics and Cyber-physical Systems for Future Maintenance: New Opportunities and Future Trends. Procedia CIRP. IEEE, 2015, 38(1), 3-7. DOI: 10.1016/j.procir.2015.08.026. ISBN 978-1-4799-7486-3. ISSN 22128271. Dostupné také z: http://linkinghub.elsevier.com/retrieve/pii/S2212827115008744

[22] MAUDER, T. Optimalizační modely s aplikacemi v organizaci výroby. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2008. 67 s. Vedoucí diplomové práce RNDr. Pavel Popela, Ph.D.

[23] MIGDALAS, Athanasios, Panos M. PARDALOS a Sverre STORY. Parallel computing in optimization. S.l.: Springer, 2012. ISBN 978-1-4613-3402-6.

[24] MOKHTARI, Hadi a Mehrdad DADGAR. Scheduling optimization of a stochastic flexible job-shop system with time-varying machine failure rate. Computers & Operations Research [online]. 2015, 61, 31-45 [cit. 2017-05-23]. DOI: 10.1016/j.cor.2015.02.014. ISSN 03050548. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0305054815000453

[25] NADERI, B. a M. ZANDIEH. Modeling and scheduling no-wait open shop problems. International Journal of Production Economics [online]. 2014, 158, 256-266 [cit. 2017-05-16]. DOI: 10.1016/j.ijpe.2014.06.011. ISSN 09255273. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/S0925527314001947

[26] NICKOLAS, S. a R. ESWARI. Solving multi-objective task scheduling for heterogeneous distributed systems using firefly algorithm. In: Fourth International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom2012) [online]. Institution of Engineering and Technology, 2012, s. 57-60 [cit. 2017-05-23]. DOI: 10.1049/cp.2012.2494. ISBN 978-1-84919-929-2. Dostupné z: http://digital-library.theiet.org/content/conferences/10.1049/cp.2012.2494

[27] POTŮČEK, Radovan. Úvod do teorie grafů. Brno: Univerzita obrany, 2013. ISBN 978-80-7231-948-0.

[28] THE MATHWORKS, INC. SimEvents® Getting Started Guide. 2015.

[29] TLUSTY, Jiri. Manufacturing processes and equipment. Upper Saddle River, NJ: Prentice-Hall, c2000. ISBN 9780201498653.

[30] TURNER, Christopher J., Windo HUTABARAT, John OYEKAN a Ashutosh TIWARI. Discrete Event Simulation and Virtual Reality Use in Industry: New Opportunities and Future Trends. IEEE Transactions on Human-Machine Systems. IEEE, 2016, 46(6), 882-894. DOI: 10.1109/THMS.2016.2596099. ISBN 978-1-4799-7486-3. ISSN 2168-2291. Dostupné také z: http://ieeexplore.ieee.org/document/7547285/

[31] UZSOY, R. Scheduling a single batch processing machine with non-identical job sizes. International Journal of Production Research [online]. 1994, 32(7), 1615-1635 [cit. 2017-05-21]. DOI: 10.1080/00207549408957026. ISSN 0020-7543. Dostupné z: http://www.tandfonline.com/doi/abs/10.1080/00207549408957026

[32] WEBER, Monika a Melanie BREDEN. Excel VBA: velká kniha řešení. Brno: Computer Press, 2007. Programování. ISBN 978-80-251-1453-7.

[33] WILLIAMS, H. P. Model building in mathematical programming. 5th ed. Hoboken, N.J.: Wiley, 2013. ISBN 978-1118443330.

[34] YANG, Xin-She. Nature-inspired metaheuristic algorithms. 2nd ed. Frome,: Luniver Press, 2010. ISBN 978-1-905986-28-6.

Seznam příloh

K práci je přiložené CD obsahující:

- elektronickou verzi diplomové práce ve formátu PDF
- GAMS: seriova_linka.gms
- GAMS: para_nolin_strid.gms
- GAMS: para_nolin.gms
- GAMS: para_lin.gms
- GAMS: obecna_linka.gms
- GAMS: fronty.gms
- GAMS: udrzby.gms
- GAMS: vizualizace.xlsm
- SimEvents: vyr_linka.slx